Thesis for the Degree of Doctor of Philosophy

Synchronous and Concurrent Transmissions for Consensus in Low-Power Wireless

Reliable and Low-Latency Autonomous Networking
for the Internet of Things

Beshr Al Nahas

Division of Networks and Systems
Department of Computer Science and Engineering
Chalmers University of Technology

Gothenburg, Sweden 2019

Synchronous and Concurrent Transmissions for Consensus in Low-Power Wireless

Reliable and Low-Latency Autonomous Networking for the Internet of Things

Beshr Al Nahas

ISBN 978-91-7905-180-8

Email beshr@chalmers.se, alnahas.beshr@gmail.com

Doktoravhandlingar vid Chalmers tekniska högskola
Ny serie nr 4647
ISSN 0346-718X
Technical Report 176D

Division of Networks and Systems
Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 (0)31-772 1000

Printed by Chalmers Reproservice
Gothenburg, Sweden 2019

Synchronous and Concurrent Transmissions for Consensus in Low-Power Wireless

Reliable and Low-Latency Autonomous Networking for the Internet of Things

Beshr Al Nahas
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

With the emergence of the Internet of Things, autonomous vehicles and the Industry 4.0, the need for dependable yet adaptive network protocols is arising. Many of these applications build their operations on distributed consensus. For example, UAVs agree on maneuvers to execute, and industrial systems agree on set-points for actuators. Moreover, such scenarios imply a dynamic network topology due to mobility and interference, for example. Many applications are mission- and safety-critical, too. Failures could cost lives or precipitate economic losses.

In this thesis, we design, implement and evaluate network protocols as a step towards enabling a low-power, adaptive and dependable ubiquitous networking that enables consensus in the Internet of Things. We make four main contributions:

- We introduce *Orchestra* that addresses the challenge of bringing TSCH (Time Slotted Channel Hopping) to dynamic networks as envisioned in the Internet of Things. In Orchestra, nodes autonomously compute their local schedules and update automatically as the topology evolves without signaling overhead. Besides, it does not require a central or distributed scheduler. Instead, it relies on the existing network stack information to maintain the schedules.
- We present A^2*: Agreement in the Air*, a system that brings distributed consensus to low-power multihop networks. A^2 introduces Synchrotron, a synchronous transmissions kernel that builds a robust mesh by exploiting the capture effect, frequency hopping with parallel channels, and link-layer security. A^2 builds on top of this layer and enables the two- and three-phase commit protocols, and services such as group membership, hopping sequence distribution, and re-keying.
- We present *Wireless Paxos*, a fault-tolerant, network-wide consensus primitive for low-power wireless networks. It is a new variant of Paxos, a widely used consensus protocol, and is specifically designed to tackle the challenges of low-power wireless networks. By utilizing concurrent transmissions, it provides a dependable low-latency consensus.
- We present *BlueFlood*, a protocol that adapts concurrent transmissions to Bluetooth. The result is fast and efficient data dissemination in multihop Bluetooth networks. Moreover, BlueFlood floods can be reliably received by off-the-shelf Bluetooth devices such as smartphones, opening new applications of concurrent transmissions and seamless integration with existing technologies.

Keywords Industrial Internet of Things, IoT, WSN, Wireless Networks, Bluetooth, TSCH, Capture Effect, Consensus, Distributed Computing.

Acknowledgements

First and foremost I want to thank my mentor and supervisor Olaf Landsiedel. His invaluable professional and scientific support were paramount to the completion of this work and my development as a researcher. While I generally enjoyed the Ph.D., his personal advice and honest discussions were essential to keeping the motivation in the tough times. I am honored to be his first Ph.D. student. I want to thank Simon Duquennoy, my friend, advisor and collaborator, from whom I learned a lot in my studies. I am grateful he continued to advise me, although he changed countries and jobs and that his current position does not formally include student supervision.

I am also indebted to my co-supervisor Philippas Tsigas for the valuable discussions and feedback. It gives me immense pleasure as well in acknowledging the support, the guidance, and the follow-up of my examiner Andrei Sabelfeld. I am grateful to Tomas Olovsson for the long discussions, the career advice and for being a caring manager. Special thanks are reserved to Agneta Nilsson for lending me her ears and for working actively to ensuring a smooth environment and high-quality Ph.D. studies.

I am honored to have Kay Römer, TU Graz, Austria, as the faculty opponent of my thesis defense. I would like to thank my grading committee, Utz Roedig, University College Cork, Ireland, Mikael Gidlund, Mid Sweden University, Sweden, and Marco Zúñiga, TU Delft, Netherlands for taking the time to review my work.

I would like to thank all the great people in the Computer Science and Engineering Department and the division of Networks and Systems for providing a friendly work atmosphere and engaging discussions. I want to thank my colleagues and friends Oliver, Valentin, Babis, Christos, Wissam, Aras, Ivan, Boel, Hannah, Fazeleh, Amir, Dimitris, Georgia, and my new friend Yahia. I greatly appreciate the insights and the friendly chats with the faculty; especially, Elad, Magnus, Marina, Ali, Katerina and Peter. I thank my friends and office mates Aljoscha, Thomas and Nasser for the bright discussions, the nice company, and for keeping the office green (Thomas).

I extend my acknowledgments to the fantastic administration with special thanks to Eva, Marianne, and Rebecca for magically providing help with all the practical matters from the important paperwork to office furniture. Special thanks go to Lasse (Lars Norén) for giving the “extra” technical care and ordering all the scientific “toys” that were crucial to perform the experiments in this work.

Finally, I could not imagine my life had I grown up in a different family. Mum and Dad, I owe it all to both of you. No words can describe how grateful I am to all the love and support you continually provide. Last but not least, I thank my love in this world: Fouz, my wonderful wife, and Luna, our little girl who lightens up the dark Swedish nights (both literally and metaphorically).

Thank you!

Beshr Al Nahas
Gothenburg, October 2019

Structure and List of Papers

This thesis follows the *collection thesis* structure commonly recommended in the technical departments of the Nordic universities. The contributions presented in this thesis have previously appeared in the manuscripts listed under the *Included Papers*. It shall be noted that the papers: A, H, I, J and the draft of B were also part of my Licentiate thesis [1].

Included Papers

A. Simon Duquennoy, **Beshr Al Nahas**, Olaf Landsiedel, Thomas Watteyne.
Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH,
Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys), 2015.

B. **Beshr Al Nahas**, Simon Duquennoy, Olaf Landsiedel.
Network-wide Consensus Utilizing the Capture Effect in Low-power Wireless Networks,
Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys), 2017.

C. Valentin Poirot, **Beshr Al Nahas**, Olaf Landsiedel.
Paxos Made Wireless: Consensus in the Air,
Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN), 2019.
This paper was nominated as a *candidate to the best paper award*.

D. **Beshr Al Nahas**, Simon Duquennoy, Olaf Landsiedel.
Concurrent Transmissions for Multi-Hop Bluetooth 5,
Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN), 2019.
This paper received the *best paper award* of the conference.

Other Papers

E. **Beshr Al Nahas**, Simon Duquennoy, Venkatraman Iyer, Thiemo Voigt.
Low-Power Listening Goes Multi-Channel,
Proceedings of the Conference Distributed Computing in Sensor Systems (DCOSS), 2014.

F. Liam McNamara, **Beshr Al Nahas**, Simon Deuqennoy, Joakim Eriksson, Thiemo Voigt.
Demo Abstract: SicsthSense - Dispersing the Cloud,
Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN), 2014.

G. Domenico De Guglielmo, **Beshr Al Nahas**, Simon Deuqennoy, Thiemo Voigt, Giuseppe Anastasi.
Analysis and experimental evaluation of IEEE 802.15.4e TSCH CSMA-CA Algorithm,
IEEE Transactions on Vehicular Technology (TVT), 2016.

H. Simon Duquennoy, Atis Elsts, **Beshr Al Nahas**, George Oikonomou.
TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation,
Proceedings of the Conference Distributed Computing in Sensor Systems (DCOSS), 2017.

I. **Beshr Al Nahas**, Olaf Landsiedel.
Competition: Towards Low-Latency, Low-Power Wireless Networking under Interference,
Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN), 2016.

J. **Beshr Al Nahas**, Olaf Landsiedel.
Competition: Towards Low-Power Wireless Networking that Survives Interference with Minimal Latency,
Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN), 2017.

K. **Beshr Al Nahas**, Olaf Landsiedel.
Competition: Aggressive Synchronous Transmissions with In-network Processing for Dependable All-to-All Communication,
Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN), 2018.

The papers I, J and K are extended abstracts of our solutions of the EWSN Dependability Competition 2016, 2017 and 2018 where we scored the third place twice and then the fourth place.

Contents

List of Figures

List of Tables

1

Introduction

An embedded cyber-physical system consists of a computer system enclosed in another bigger system to serve the operation of the encompassing system by utilizing sensors, actuators and data processing to make informed operations that potentially affect the physical environment around it. In other words, cyber-physical systems are

> *physical and engineered systems whose operations are monitored, controlled, coordinated, and integrated by a computing and communicating core [116].*

Cyber-physical systems are embedded around us to ease our everyday life, *e.g.*, in elevators, cars, and airplanes. For example, the autopilot system used in aviation is a cyber-physical system. It depends on the feedback of sensors measuring the relevant environment's and system's parameters to fly the airplane. Flying the plane changes the airflow around; thus, feedback is necessary to maintain its operation. Such a complex system encompasses many coordinating subsystems. For example, a car embeds hundreds of control units that are connected through a wired network [104]. Each of these units is responsible for a function, *e.g.*, controlling and monitoring speed, brakes, engine temperature or non-critical functions like entertainment. One side effect is that a car has over 1 km of wires to connect these subsystems. This complicates both manufacturing and maintenance. Therefore, car manufacturers aim to convert these installations to wireless [150].

Moreover, networked objects are everywhere in our lives. We have become so used to being always online that we feel nervous when we are not [142]. Our laptops and cell phones are always connected to the Internet, and even our homes are connected, too: Alarm systems, security cameras and the smart grid which feeds our homes with electricity and updates the utility company with our consumption and the grid status. Industrial giants like Cisco and Ericsson predict a growing connectivity and project 22 billion devices to be connected by 2024, including roughly 18 billion short-range IoT devices [22].

If this connectivity trend lives up to the predictions, a variety of appliances will be connected either to each other's only or the Internet – in what is called the Internet of Things (IoT) – to enable remote control and automatic actions. Nowadays, the hype is about autonomous and coordinating cars to enhance safety and reduce congestion on roads. However, with this arises the need for new connectivity methods to enable car-to-car and car-to-infrastructure communication to support a safer and smoother mobility experience.

Apart from everyday objects, industrial actors aim to enhance the automation of their factories with sensor networks and connections to cloud services to, for example, predict failures and trigger maintenance procedures automatically, as envisioned by Industry 4.0 [49] and the Industrial Internet of Things (IIoT). All of the aforementioned scenarios require connectivity, and with the envisioned higher degree of connectivity, *e.g.*, that includes moving parts, we cannot imagine wires running all over the place. Therefore, we turn to wireless solutions.

Having a wireless solution entails the lack of access to a wired energy source; thus, we have a limited power source from a battery or an energy harvester, *e.g.*, a solar cell. Moreover, with wireless solutions, we expect data losses and communication outages, due to interference from other technologies. However, not only this is mildly annoying when it happens to us when we are, for example, surfing the internet and having to wait for a website to open, it is probably unbearable if we try to operate a wireless lamp and have to press the button again and again as the communication between the button and the lamp is unreliable. Besides, such a data loss could have a catastrophic impact if it happens in a critical system, as a wireless brake system in a car, for example.

In this thesis, we design, implement and evaluate dependable network protocols that cope with the challenge of achieving a dependable operation over low-power and lossy wireless links with limited energy sources and processing power.

1.1 Target Applications: Classification, Requirements and Challenges

In this section we first illustrate our problem statement and our approach, then we discuss the industrial applications classification and their characteristics. Next, we introduce the requirements of target applications, and the challenges we need to overcome to achieve these requirements.

1.1.1 Problem Statement and Approach

With the emergence of the Internet of Things, autonomous vehicles and the Industry 4.0, the need for dependable yet adaptive network protocols is arising. Many of these applications build their operations on distributed consensus. For example, networked cooperative robots and UAVs agree on maneuvers to execute, and industrial control systems agree on set-points for actuators. Many applications are mission- and safety-critical, too. Failures could cost lives or precipitate economic losses.

Any wireless network connecting mission-critical devices must be dependable, and often energy-efficient, as many devices are battery-powered and we expect them to last for years. Moreover, application scenarios in the Internet of Things imply a dynamic environment with a changing network topology due to mobility and interference, for example. Thus, the network protocol shall be adaptive and self-organizing as well, to allow for dependable autonomous operations, as many applications cannot afford to stop and wait for external (re)configuration.

In this thesis, we use experimental computer science methods: We design, implement and evaluate network protocols as a step towards enabling such challenging ubiquitous connectivity in the Internet of Things. We contribute the source code of our main protocols to the community as a step towards enabling ubiquitous connectivity in the Internet of Things.

1.1.2 Applications Classification

IETF RFC 5673 [111] classifies industrial applications into three application categories. Namely, safety, control, and monitoring. These categories have six criticality classes ranging from the always-critical to the never-critical operations. In-time delivery is more paramount in the lower classes, *i.e.*, with the higher criticality and jitter is as important as latency for achieving a stable control [111]. Moreover, Åkerberg *et al.* [2] and others [88, 90, 123] highlight specific characteristics of such industrial applications. We summarize them in the following, and we base primarily on the classification in the RFC 5673 [111]:

- **Safety Category**
  - *Class 0: Emergency action – Always critical*: Dormant safety-critical operations that activate upon failures, *e.g.*, fire alarm and fire control.
- **Control Category**
  - *Class 1: Closed-loop regulatory control – Often critical*: Factory automation systems such as robotic arms that place parts on moving bands, or process automation systems that automatically set the operating parameters to control an industrial process.

Table 1.1: Typical requirements for industrial and home automation applications. *We notice that typical closed-loop control systems require fast delivery, e.g., in tens of milliseconds range and a low loss-rate. We mainly focus on the slower categories, but keep an eye on the fast ones as well in our discussions.*

Class	*Domain*	*Update Interval*	*Loss Rate*
4, 5	Monitoring, Alerting and Logging	100 – 1000 ms	10^{-2}
3, 4, 5	Building Automation	500 ms – seconds	10^{-3}
2, 3	Open-loop and Closed-loop Supervisory Control	10 – 100 ms	10^{-4}
1, 2	Process Automation	10 – 1000 ms	10^{-5}
1	Factory Automation	500 μs – 100 ms	10^{-9}

  - *Class 2: Closed-loop supervisory control – Usually non-critical*: Supervisory systems that report the status of the closed-loop control. Such supervisory control systems usually operate with a human setting a control point and monitoring from a control room.
  - *Class 3: Open-loop control – Human in the loop*: Operator-controlled systems where a human controls the actuator and monitors the system reaction.

– **Monitoring Category**
  - *Class 4: Alerting – Short-term operational effect*: Monitoring and supervisory systems that track system status to detect machinery problems and require event-based maintenance, for example.
  - *Class 5: Logging and downloading / uploading – No immediate operational consequence*: Long-term logging and diagnostics operations that can be used for recording history, preventive maintenance, and interactive fault investigation, for example.

It shall be noted that *home and building automation systems* can be classified under classes 3, 4 or 5; except for emergency systems such as fire alarms.

This classification can be linked to specific requirements in terms of acceptable message loss rate and expected update intervals, end-to-end. The acceptable end-to-end message delivery delay is usually in the order of the update interval, and the rule of thumb is that each measurement shall arrive before the deadline of the next one. Otherwise, the delayed measurement becomes useless. Table 1.1 summarizes these requirements based on the work in [2, 88, 90, 98, 123].

In the following section, we discuss the requirements of a network protocol for the targeted applications. Later, we discuss the challenges of achieving these requirements.

1.1.3 Requirements

We notice that typical closed-loop control systems require fast delivery, *e.g.*, in tens of milliseconds range and a low loss-rate between 10^{-4} and 10^{-9}. In this thesis, we mainly focus on the slower categories that can survive a delay of tens to hundreds of milliseconds and a loss rate between 10^{-3} and 10^{-5}. We keep the most demanding applications with 10^{-9} loss rate for future work, as guaranteeing a loss rate limited to 10^{-3} proves to be challenging with the available low-power multihop wireless technologies. However, much of the solutions we discuss in this thesis can be adapted to faster and more reliable wireless technologies. We can summarize the requirements of the *network protocol* that enables these applications in the following [46, 88, 132]:

Low Power One of the main motivations for deploying wireless systems is getting rid of wired connections, to ease installations and save costs. Therefore, in many cases, where an electrical grid connection is not available, the target systems shall be ultra low-power [132]. It should be able to operate for five to ten years on batteries [153] or energy harvesters, as it is not desirable to replace batteries more often. Since the radio is one of the most energy-consuming components in such small devices, we are interested in minimizing its energy use. However, this might not be required for all systems, especially those with a very high update rate, as the radio is not the energy-critical component anymore, but the sensing and sampling components. Thus, the enclosing device shall have an electrical power connection, which can power the wireless communication system as well [2].

Low Latency The network protocol shall provide timely information delivery as dated information might lose its worth. For example, a smoke detector needs to deliver a warning before a deadline, and industrial control applications require sensor readings to be delivered to the final destination before a subsequent update.

Technology Independent The network protocol shall not depend on specific physical layer support and can run on top of a variety of commercially available low-power wireless standards, *e.g.*, IEEE 802.15.4 and Bluetooth. The key is not to be locked in a proprietary technology [132].

Supports Rapid Network-wide Consistency and Consensus Many applications in low-power wireless networks build their operation on consensus: For example, networked cooperative robots and UAVs agree on maneuvers to execute [7], wireless closed-loop control applications such as adaptive tunnel lighting [21] or industrial plants [103, 105] agree on set-points for actuators. These application scenarios exhibit key differences when compared to traditional data collection or dissemination in wireless sensor networks: They de-

mand primitives for network-wide consensus at low-latency and highly reliable data delivery with robustness to interference and channel dynamics [3].

Dependable Since the applications classified at levels 0 to 4 are usually mission- or safety-critical, they shall be *dependable* in order to avoid tragic life or economic losses. According to Laprie [86] and Avizienis *et al.* [8], dependability is defined as *the ability to deliver a service that can justifiably be trusted*, or *the ability to avoid service failures that are more frequent and more severe than is acceptable*. Dependability entails the following attributes [8, 86]:

- *availability:* readiness for correct service.
- *reliability:* continuity of correct service. For the network protocol, this entails that the end-to-end packet loss shall suit both non-critical and mission-critical applications, as data loss is undesirable at best and could be disastrous in critical scenarios. For the consensus requirement this entails that the system continues to achieve progress and does not abort transactions often due to failures.
- *safety:* absence of catastrophic consequences on the users and the environment. For example, if an inconsistency causes a catastrophe, then the consensus process shall avoid them.
- *maintainability:* ability to undergo modifications and repairs.
- *integrity:* absence of improper system alterations. For example, the corruption of a packet's contents shall be detected.

In this thesis, we focus on the first three aspects of dependability: Availability, reliability, and safety.

Flexible The network protocol shall be able to satisfy a variety of often dynamic application requirements; such that it *self-forms* a network on its own without relying on external components or manual configurations, and it *self-fixes* the network and copes with link dynamics and node failures, to achieve a dependable and continous operation, as failures cannot be avoided.

Finally, we shall note that *security and privacy* are key requirements [132], but they are out of the scope of this thesis.

In summary, we desire a network protocol that is (i) low power, (ii) low latency, (iii) not tied to a specific wireless standard, (iv) provides network-wide consistency and consensus, (v) dependable, and (vi) flexible (self-forming and self-fixing). We note that achieving each of these requirements alone is demanding, and realizing a protocol that achieves them together is more challenging, as we discuss next.

1.1.4 Challenges

The requirements introduced in §1.1.3 are challenging to achieve due to the nature of the application scenarios. First of all, the low-power requirement neces-

sitates the use of resource-constrained platforms, which we demand to achieve a timely and reliable performance. For the same reason – energy efficiency –, we need to use low-power wireless communications that are characterized by variable quality links. This, in turn, stipulates the need for a multihop mesh networking topology to cover wide areas with low-power wireless. Finally, distributed consensus is a hard problem even when implemented on the more capable devices and stable networks such as those found in data centers. Consensus becomes even more challenging to realize under these constraints as faults are inevitable in such resource-constrained low-power wireless networks. In the following, we detail the aforementioned challenges.

Resource-constrained Embedded Platforms To satisfy the low-power requirement, a typical computing platform in low-power wireless usually features a small form factor and limited processing, memory, and storage components [132]. This stipulates the use of a simple protocol logic and prohibits complex operations. Such complex operations can be effective on other, more powerful, platforms but are prohibitively expensive on these low-power devices. For example, to enhance the robustness of the network we can use complex data encoding schemes with forward error correction (FEC) [143], *e.g.*, LDPC and Turbo Codes. In practice, the implementation of such techniques would consume a major part of the available memory and bandwidth on such devices. Moreover, the execution of these operations would take a relatively long time compared to acceptable latencies, *e.g.*, tens to hundreds of milliseconds on such limited processors [154].

A popular platform in low-power wireless research is TelosB [141] which features a Chipcon CC2420 250 Kbps radio operating in the 2.4 GHz and compatible with IEEE 802.15.4. It features a Texas Instruments MSP430 8 bit micro-controller operating at 4 MHz with 10 KB RAM and 48 KB flash for program storage. While this platform is more than a decade old and superseded by more powerful platforms, its moderate capabilities can fit on a tiny SoC with the modern chip manufacturing processes. Therefore, it is still relevant when considering the *Smart Dust* vision [144] of one cubic millimeter sensing platforms. There are newer more powerful platforms, such as these based on the ARM Cortex M0 to M4 32 bit MCUs with operating frequencies up to 64 MHz, up-to 64 KB memory, 256 KB program storage and on-SoC radio supporting both 802.15.4 and Bluetooth. These can use frequency scaling to operate in low-power modes while being able to boost the frequency when the application needs more complex processing. Thus, in both platform categories, it is desirable to have a simple protocol logic and avoid complex operations to save power, while at the same time duty-cycling the radio and keeping it turned off as long as possible.

Wireless Links are Variable Low-power wireless communications are challenging in several ways:

- *Unreliable Links*: The wireless links are unreliable due to the noise coming from the environment, electrical machines and radio interference from other devices using the same radio frequency [52, 147]. Moreover, cross-channel interference from adjacent channels causes a significant packet loss rate [10];
- *Asymmetric Links*: The wireless links are not always symmetric; especially when the link quality is medium or transitional, *i.e.*, neither very high nor very low. For such links, we cannot conclude that a node A can receive from B even if B can receive from A. Further, the link asymmetry is not necessarily linked with distance, nor is it always persistent [10]; and,
- *Challenging Link Dynamics*: The nature of radio wave propagation and multi-path fading cause challenging link dynamics that affect the signal strength and packet reception rate in different ways. Multi-path effects can either strengthen or weaken the link quality depending on a number of parameters; namely, the used frequency, the objects standing/moving in the wireless path and the location of the transceiver [10, 145].

The result is that low-power wireless links are hard to predict and present a continuously changing state, both spatially and temporally [10, 159].

Multihop Mesh Networks When it comes to low-power wireless communication technologies, we have two main classes: (i) The long-range (1 – 10 km), *e.g.*, LoRa, 802.15.4-sub-GHz, 5G-NBIoT and SigFox, and (ii) the short-range (10 – 100 m) technologies, *e.g.*, 802.15.4-2.4 GHz and Bluetooth.

Since long-range technologies can cover a whole factory or residential area, for example, they offer a simple network topology, *e.g.*, a star topology. It is, therefore, tempting to consider that they solve all the connectivity problems. In fact, the longer the range, the less spectrum-efficiency a connectivity solution exhibits. Consider, for example, the congested WiFi spectrum where one sees all neighbors' WiFi networks. A similar problem will materialize for the long-range technologies when everybody starts using them; except, the interference range is longer, *e.g.*, several kilometers. Specifically, the following limitations materialize in long-range technologies:

- *The channel capacity is limited*: The long-range means potentially more interferers. When using a licensed spectrum, *e.g.*, in 5G-NBIoT, the devices will be competing for spectrum which has to cover a long-range, and only a limited number will be supported by one base-station; thus, requiring a possibly expensive deployment from the telecom provider. Similarly, the unlicensed sub-GHz ISM band will eventually be crowded; thus, resulting in collisions;

- *Low data rate*: To achieve a long-range at a low energy budget, a communication standard enhances the SNR at the receiver by narrowing its bandwidth [73]. For example, 5G-NBIoT uses 200 KHz channels. The other alternative is to use Spread Spectrum techniques [73], *i.e.*, the transceiver uses wider channels but uses a larger spreading factor to achieve a higher redundancy. For example, LoRa channel bandwidth is between 125 and 500 KHz. Both strategies result in low data rate. For example, LoRa has data rates between 0.18 and 27 Kbps;
- *High latency and long packet transmission time*: With a low data rate, the packet takes a longer time to be transmitted; thus, it is more susceptible to interference. Besides, the minimum latency becomes higher. For example, a LoRa packet can last up to several seconds. Moreover, this increases the energy expenditure of the packet [20]; thus, making packet losses even costlier;
- *Large delay and limited channel utilization*: In order to ensure a fair access policy of the long-range ISM sub-GHz wireless medium, the channel usage is regulated: The device can have a maximum of 1% duty-cycle, and has an imposed channel-off time, where it cannot send before waiting a specific time, such that at any time window, the duty cycle does not exceed the 1% limit [130]. For example, if a packet transmission takes 1 s with LoRa in the 868 MHz band, it shall avoid using the same band for 99 seconds;
- *Complex configuration*: The unlicensed technologies, such as LoRa and 802.15.4-sub-GHz, have tens of parameters to adjust, and choosing the wrong mix can severely hurt the performance. For example, Bor and Roedig [14] show that LoRa can have thousands of different combinations of parameters' settings, and choosing the wrong combination can increase the energy budget for a given link quality by a factor of 100; and,
- *Fault intolerance*: The typical star topology with a single base station is not fault-tolerant. Instead, several base stations shall be used with a fault-tolerant mechanism for take over and rejoin. This presents a deviation from the sought simplicity of the star topology. Moreover, this might be a costly solution for licensed technologies such as 5G-NBIoT [47].

The alternative is the category of short-range technologies such as 802.15.4-2.4 GHz and Bluetooth. These low-power wireless devices have a limited transmission range, especially, when operating indoors, due to the limited transmission power possible at the available energy budget. For example, typical transmission power for Bluetooth 5 is about 1 mW, equaling 0 dBm, while WiFi devices can send at 100 mW (20 dBm) when operating in the 2.4 GHz band. Therefore, to cover larger areas while benefiting from the limited interference range of the short-range technologies, we can extend the range by

organizing the network as a multihop mesh. Thus, nodes cannot reach the final destination directly, but rather have to pass messages through intermediate nodes. Compared to the star-topology common with classic wireless networks, *e.g.*, mobile networks, WiFi hotspots, and legacy Bluetooth devices, this poses several challenges:

- end-to-end connectivity is not simple anymore, as the network protocol needs to maintain routes or consider stateless approaches such as flooding;
- nodes consume more energy and have a higher processing burden as intermediate nodes shall operate as routers or forwarders to maintain network connectivity; and,
- it is complex to achieve high end-to-end reliability as it depends on the continuously changing quality of the forwarding links.

It shall be noted that it is possible to configure some of the long-range technologies, *e.g.*, LoRa, to cover a shorter range with a higher bitrate. Thus, offering a flexible alternative, for the overhead of the reduced energy efficiency and software complexity. Nevertheless, the need for multihop mesh further complicates achieving the energy, latency and reliability requirements *c.f.*, §1.1.3; especially, when realizing a complex service such as the network-wide consensus.

Dependable Consensus in Presence of Network Partitions and Failures Dependability is threatened by the occurrence of internal or external faults, *e.g.*, a node's hardware fails, or external radio interference occurs. These faults are more pronounced when considering constrained and low power devices connected through volatile wireless links in a multihop setting. Such faults may cause a failure, *e.g.*, a packet loss, or even a network partition. The consensus problem is a well-studied problem in the classic distributed systems literature. Since node failures and network partitions are common in low-power wireless systems, achieving consensus is particularly challenging in such distributed systems where faults can happen, as illustrated by two important results: The FLP impossibility result [43], and later, the CAP theorem [48]. FLP shows that it is impossible to achieve distributed consensus if only one process can crash in an asynchronous setting where one cannot distinguish a failed process (or a message loss) from a process that is simply taking a long time to reply. The other important theorem (CAP) can be summarized in layman terms as follows: It is impossible to maintain a consistent (C) and always available (A) consensus that can sustain network partitions (P) in an asynchronous setting, *e.g.*, if even one participant can fail and stop working, or if messages can be lost without detection. An undetected message loss is another symptom of a process failure if the receiver cannot distinguish between the two

cases: The message was never sent and the process crashed, or the process is alive and the message was lost or delayed due to retransmissions.

To this end, the two-phase commit protocol (2PC) [53] is a relatively simple protocol that was designed for distributed consensus on atomic commit. It has a known shortcoming of inability to reliably sustain failures in certain conditions without blocking and limiting the system's availability [53]. Despite that, it was adopted in distributed databases, which have a common size of several servers per cluster. Still, with this limitation and its simple design, this protocol requires at least three group communication rounds: (i) disseminate a proposal (one-to-all), (ii) collect votes (all-to-one), and (iii) disseminate a decision of unanimous accept, otherwise, abort (one-to-all). While this is a relatively simple protocol, applying it in the low-power wireless networks context is challenging: The sensor networks' sizes range from tens to hundreds. Therefore, 2PC group communication nature and its several rounds of communication have a high overhead; especially, when using classic unicast-based protocols. Further, lossy wireless links are problematic for the execution of this protocol, as it could block frequently and hinder progress.

Later protocols, like Paxos [82], that accomplish consistency in such asynchronous settings, assume *eventual synchrony*: Messages *eventually* arrive at their destinations, and node failures are masked by assuming stable storage of protocol state that a recovering node can use to resume execution where it stopped. Thus, the failure merely appears as delayed processing to other nodes. This makes the system *eventually consistent*, where it has a transient inconsistent period while the messages get delivered, a crashed node restarts and resumes operation, or a replacement gets elected. In practice, such protocols are more complex than the two-phase commit protocol as they solve the general problem of consensus, *e.g.*, they have to deal with conflicting proposals and provide a stronger failure tolerance. Thus, they are complex to understand and correctly implement [107]. We provide more details on consensus next in §1.2.1.

1.2 Background

In this section, we review the necessary technical background that we use in the rest of the chapter. We discuss consensus and its main protocols: Two-phase commit [53], three-phase commit [133] and Paxos [81]. Later, we overview the radio physical layers and MAC protocols that we build our work on top; namely, IEEE 802.15.4, TSCH [61], Glossy [42], and Bluetooth 5 [12]. Finally, we discuss the radio phenomena that enable receiving one of several concurrently transmitted packets, under certain conditions, that otherwise result in

a collision. Later, §1.3 provides a deeper discussion of the related work in the low-power wireless communications and the Internet of Things.

1.2.1 Consensus

The consensus is the problem of reaching agreement among several processes about a proposal or different proposals, *i.e.*, to accept or decline it after a finite time of execution. Solving consensus is key to solving many problems in distributed systems, such as atomic commit, leader election, and group membership. It has been shown that any of these problems reduce to the consensus problem [79]. Thus, it is possible to derive a solution to one problem from the solution of another [26]. A correct solution of the consensus problem shall have the following properties [26, 79]:

- *Validity*: the consensus result is one of the proposed values;
- *Agreement*: all correct processes decide on the same value;
- *Termination*: every correct process decides in a bounded time; and,
- *Integrity*: a process cannot change its decision or decide multiple times.

Achieving consensus becomes challenging when faults may occur, that is, when communication is lossy and processes may crash.

Two widely used, yet simple consensus protocols are *two-phase commit (2PC)* [53] and *three-phase commit (3PC)* [133]. However, both are vulnerable to faults, with 2PC responding by blocking in some cases, and 3PC yielding an inconsistent outcome in other cases. We later highlight Paxos [82], which achieves fault-tolerant consensus if the majority of nodes are non-faulty. The faulty nodes exhibit an eventually consistent behavior, where they will have a stale state until they receive further updates. We review these protocols next and discuss their respective properties and limitations.

Two-Phase Commit (2PC) The two-phase commit protocol solves the problem of *transaction commit*; *i.e.*, one transaction manager proposes a transaction and later decides to either *commit* or *abort* it atomically. The protocol assumes the existence of one static transaction manager, or coordinator, and a set of participants, or cohort. As the name suggests, 2PC works in two phases, Proposal Voting and Decision: (i) *Proposal Voting:* the coordinator broadcasts a proposal to the cohort, each member replies with its vote, *yes* or *no*; (ii) *Decision:* the coordinator decides to *commit* if the vote is *yes* unanimously; otherwise it decides to *abort*. It then broadcasts the decision to the cohort that will commit or abort upon receiving the decision message.

2PC is simple to realize but has the major limitation of being a *blocking protocol*. Whenever a node fails, other nodes will be waiting for its next message or acknowledgment indefinitely, *i.e.*, the protocol may not terminate. Recovery

schemes can be considered but fall short when it comes to handling two or more failing nodes [54]. In particular, if the coordinator and a participant both fail during the second phase, other nodes might still be in *uncertain* state, *i.e.*, have voted *yes* but have not heard the decision from the coordinator. If all remaining nodes are *uncertain*, they are unable to make a safe decision as they do not know whether the failed nodes had committed or aborted before failing. Thus, the uncertain nodes block until the failed nodes are online again.

Three-Phase Commit (3PC) Three-phase commit mitigates the above limitations by decoupling *decision* from *commit*. This is done with an additional *pre-commit* phase between the two phases of 2PC. The three phases are as follows: (i) *Proposal Voting:* same as in 2PC; (ii) *Pre-Commit (or abort):* the coordinator and participants decide as in 2PC, but no commit is applied (abort is applied immediately); (iii) *Do Commit:* participants finally commit. The additional phase guarantees that if any node is *uncertain*, then no node has proceeded to commit.

The protocol is non-blocking in the case of a single participant failing: Remaining nodes time out and recover independently (*commit* or *abort*). 3PC can also handle the failure of the coordinator and multiple nodes, by using a recovery scheme. Nodes will then enter a *termination protocol*: They will communicate and unanimously agree to *commit*, *abort*, or take over the coordination role and resume operation. In the more challenging case of a network partition, 3PC is, however, unable to maintain consistency.

Paxos is a fault-tolerant protocol for consensus [82]. It assumes an asynchronous, non-Byzantine system with crash-recovery; *i.e.*, it handles (i) both process crash and recovery (persistent storage is needed), but not misbehaving nodes or transient faults; (ii) delayed or dropped messages, but not corrupted messages; and (iii) network segmentation. The protocol guarantees a correct consensus if the asynchronous network becomes eventually synchronous; *i.e.*, messages get delivered eventually, and nodes fail and restart with access to permanent storage. Moreover, it is non-blocking if the majority of nodes are available.

A node can act as either a *Proposer*: It proposes a value to agree on and acts as a coordinator, or an *Acceptor*: It votes on proposers' requests. Unlike 2PC and 3PC, where at most one coordinator must be present, Paxos tolerates multiple proposers, at the cost of impeding the progress of the agreement.

The protocol consists of two phases: The *Prepare* phase and the *Accept* phase.

1. **Prepare Phase**

a. A proposer starts by broadcasting a *Prepare(n)* request that includes a unique proposal number n.

b. Upon reception of a *Prepare(n)* request, an acceptor saves the highest proposal number *minProposal* it heard so far. The idea is that *minProposal* represents the minimum proposal number that can be accepted, as proposals with higher numbers have priority. The acceptor replies with both the last *accepted proposal*, if any has been accepted so far, and the corresponding *accepted value*.

2. **Accept Phase**

a. Upon hearing from a majority of acceptors, the proposer *adopts* the value with the highest proposal number, if any. Thus, at most one value can be chosen. The proposer switches to the *Accept Phase* and sends an *Accept(n,V)* request to all acceptors.

b. Upon receiving an *Accept(n,V)*, an acceptor *accepts* the value V if and only if the proposal number n is higher or equal to the proposal number the process has prepared for: *minProposal*. Then it replies to the request by including the highest proposal heard (*minProposal*).

c. Upon receiving at least one reply with $minProposal > n$, the proposer knows that its value has been *rejected*. This also means at least one other proposer is present, and the process can either restart the protocol with a higher proposal number $\acute{n}$ to compete, or let the other proposer win. If the proposer received a majority of replies with no rejection, the value is *chosen*. The competition is on the premise that the first proposer that succeeds to achieve a majority would have its value chosen. However, once a value gets chosen by the majority, any later competition would adopt the already chosen value that the proposer receives in the step 1.2.1.

Using *minProposal* ensures that only the most recent proposal can be accepted and the data returned at step 1.2.1 ensures that at most one value can be chosen per one Paxos execution. To accept more values, we have to execute new Paxos instances, as done in MultiPaxos.

Paxos vs. Two-phase Commit (2PC) While the two-phase commit protocol is designed to solve the problem of *transaction commit*, Paxos solves the general consensus problem, *i.e.*, agreeing on one of multiple competing proposals from multiple proposers. Therefore, it is possible to express the two-phase commit protocol as a special case of Paxos with a single proposer that blocks on failures [54]. However, a practical solution would implement multiple proposers for fault tolerance.

Next, we discuss the low-power communication standards and relevant protocols.

1.2.2 Low-Power Wireless Protocols

ZigBee/IEEE 802.15.4 and Bluetooth Low Energy (BLE) are today's widespread technologies for low-power wireless communication in the unlicensed 2.4 GHz spectrum. Each of them was initially designed for unique and distinct goals: While Bluetooth traditionally targets low-range single-hop communication with a bitrate suitable for *e.g.*, wearable and multimedia applications, ZigBee targets longer ranges and reliable multihop communication with a lower bitrate suitable for *e.g.*, home automation applications or industrial control. To this end, the IEEE 802.15.4 standard introduces a physical layer in the 2.4 GHz band that utilizes O-QPSK modulation and DSSS for forward error correction (FEC): The PHY layer groups every 4 bits to make one PHY symbol and encodes it using 32 PHY signals or *chips* — each is a half-sine that represents either a logical 0 or 1. With a chip rate of 2 M chips per second, it supports a bitrate of 250 Kbps in 16 RF channels of 5 MHz. It offers a packet size of up to 127 bytes.

On the other hand, both Bluetooth and 802.15.4 in sub-GHz use variants of FSK modulation. BLE 4 uses GFSK and the latter uses 2-FSK — both modulation schemes represent bits 0 and 1 by using a $\pm F$ frequency shift from the central frequency. BLE 4 offers a bitrate of 1 Mbps in 40 channels with a bandwidth of 2 MHz each without FEC and supports packets with PDU up to 39 bytes. Overall, the design choices of the narrower channels, a simpler modulation scheme and the lack of DSSS make Bluetooth the less robust communication scheme of the two. Next, we discuss how the recent Bluetooth 5 changes this.

Bluetooth 5 With the widespread availability of Bluetooth and an estimated number of 10 billion Bluetooth devices sold, there is an increasing interest to use Bluetooth beyond the originally targeted domain of low-range, single-hop communication. Hence, the recent Bluetooth 5 standard [151] introduces (i) new long-range communication modes and (ii) supports longer packets up to 255 bytes.

The physical layer of Bluetooth 5 supports four PHY modes: (i) Two modes without forward error correction (FEC): A new, 2 Mbps mode in addition to the backward-compatible 1 Mbps, and (ii) two new long-range modes that utilize FEC driven by a convolutional code: 500 Kbps and 125 Kbps. These coded modes support up to 4× longer range when compared to the uncoded 1 Mbps, outdoors. We note selected low-level details: (i) The different modes have different preamble lengths: One byte for 1 M, two bytes for 2 M and ten bytes for the coded modes 500 K and 125 K; (ii) the two coded modes 500 K and 125 K always transmit the header with FEC 1:8, and only afterward change the coding rate to FEC 1:2 for the 500 K mode; and (iii) all modes share a

Table 1.2: Bluetooth 5 and IEEE 802.15.4: PHY parameters and modes. *Note that: (a) in Bluetooth, each bit is encoded using 1, 2 or 8 symbols depending on FEC; (b) Bluetooth coded modes 500 K and 125 K use the 1 M PHY mode beneath, and (c) IEEE 802.15.4 uses a different terminology: one symbol represents 4 bits and is encoded using 32 chips — a chip is the PHY layer signal that represents a logical 0 or 1. τ stands for period.*

Modulation	*Bitrate* [*bps*]	*Symbol rate* [Symbol/s]	*Symbol* τ [μs]	*bit* τ [μs]	*FEC* ratio	*Preamble* [byte]
Bluetooth 5:						
GFSK	2 M	2 M	0.5	0.5	-	2
GFSK	1 M	1 M	1	1	-	1
GFSK	500 K	1 M	1	2	1:2	10
GFSK	125 K	1 M	1	8	1:8	10
Modulation	*Bitrate*	*Chip rate*	*Chip* τ	*Symbol* τ	*FEC*	*Preamble*
IEEE 802.15.4 @ 2.4 GHz:						
OQPSK	250 K	2 M	0.5	16	1:8	4

symbol rate of 1 M except for the 2 M mode. Table 1.2 summarizes the operation modes. When compared to 802.15.4, the physical layer of Bluetooth 5 still maintains the narrow channels of 2 MHz and does not employ DSSS. Nonetheless, the standard has the potential to be an enabler for IoT applications with a performance in terms of range, reliability, and energy-efficiency comparable to 802.15.4.

Bluetooth Mesh part of the Bluetooth 5 standard, introduces multihop communication to Bluetooth [57]: Bluetooth Mesh follows a publish/subscribe paradigm where messages are flooded in the network so that all subscribers can receive them. Thus, Bluetooth Mesh does not employ routing nor does it maintain paths in the network. To reduce the burden on battery-powered devices, forwarding of messages in a Bluetooth Mesh is commonly handled by mains-powered devices. In recent studies with always-on, *i.e.*, mains-powered, nodes as the backbone, Bluetooth Mesh reaches a reliability of above 99% both in simulation [96] and experiments [131], and latencies of 200 milliseconds, in networks of up to 6 hops with a payload of 16 bytes [131].

Because Bluetooth Mesh employs flooding, it differs strongly from established mesh and routing protocols in 802.15.4 such as CTP [50] or RPL [137].

IEEE 802.15.4-2015 Time Slotted Channel Hopping (TSCH) is a MAC layer specified in IEEE 802.15.4-2015 [61], with a design inherited from

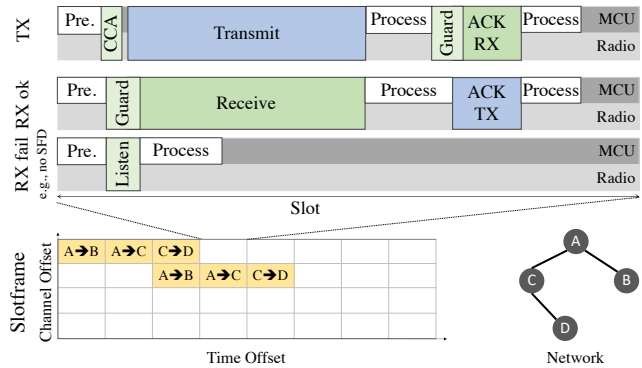


Figure 1.1: Diagram of a TSCH timeslot and example slotframe. A timeslot is typically 10 ms long and fits both frame reception or transmission, acknowledgment and processing. The radio is switched on only when listening, receiving or transmitting, and turned off otherwise to save power. Slots are grouped in slotframes which repeat periodically.

WirelessHART and ISA100.11a. TSCH builds a synchronized mesh network. Nodes join the network upon hearing a beacon from another. Subsequently, they may broadcast the beacon to reach further nodes. TSCH uses both time division (TDMA) and frequency diversity for coordinating the nodes' multiple access to the radio medium. Time is cut into timeslots which are grouped into periodic slotframes (as illustrated in Figure 1.1). Slots can be dedicated or shared, *i.e.*, contention-free or contention-based with CSMA back-off.

Time synchronization trickles from the coordinator down to leaf nodes along a Directed Acyclic Graph (DAG) structure. Nodes update their synchronization relative to their *time source parent* every time they receive a packet from it.

TSCH networks use channel hopping: The same slot in the schedule translates into a different frequency at each iteration of the slotframe. The result is that successive packets exchanged between neighbor nodes are communicated on different frequencies. In case a transmission fails because of external interference or multi-path fading, its retransmission happens on a different frequency, often with a better probability of succeeding than using the same frequency again [145].

How the communication schedule in the TSCH network is built and maintained is out of the scope of the established standards.

RPL is the standard IPv6 routing protocol for low-power and lossy networks (LLNs) [137]. It was built specifically to support the requirements of LLNs which exhibit special characteristics such as: Limited energy, limited processing

capabilities, and highly dynamic topologies (because of link instability and node failures).

RPL builds a directed acyclic graph (DAG) representation of the network. A DAG is a tree-like structure. It has a single root node that has no parents and usually represents a border router, and each node can have multiple parents. Thus, DAGs support redundancy naturally.

RPL supports three modes of traffic [137]:

- Point-to-point (*i.e.*, unicast);
- One-to-Many (*i.e.*, multicast) such as downlink traffic from root to children; and,
- Many-to-One (*i.e.*, converge-cast) such as uplink traffic from children to root.

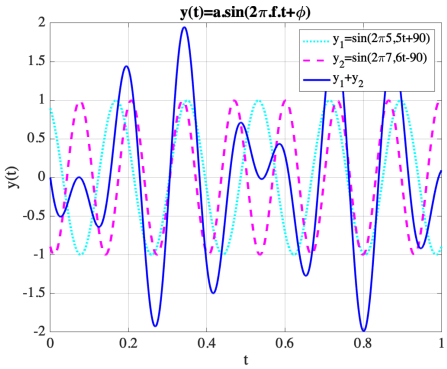
It shall be noted that the One-to-Many and Many-to-One modes are usually implemented using unicast.

RPL can detect loops, and dynamically restore network connectivity after node or link failures. If a node can reach neither its parent nor any backup (in the up direction), it initiates a local repair to find another parent. Local repair is simply done by broadcasting a DAG information solicitation (DIS) message. Neighboring nodes reply to this by sending DAG information object messages (DIO) back, enabling the requester to choose the best available parent. This might result in a sub-optimal path for this part of the DAG, but it does not require a network-wide routing update. However, the root node can trigger a global repair, which rebuilds the whole DAG from scratch; yielding a more optimal DAG at the cost of the increased routing information traffic. Moreover, RPL employs a data-path validation mechanism to facilitate detection of possible loops by adding direction flags, *e.g.*, up or down, to routed packets. When a router detects a loop while processing these flags, it discards the data packet and initiates local repair.

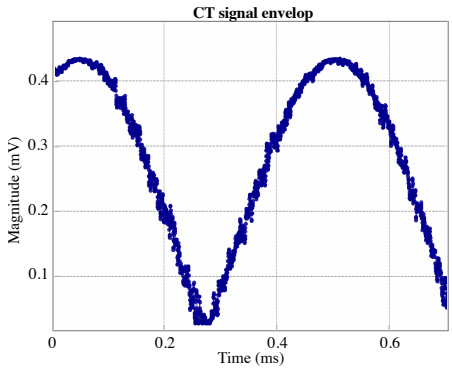
1.2.3 Concurrent Transmissions and the Capture Effect

In this section, we discuss concurrent transmissions (CT) in a generic context that applies to both IEEE 802.15.4 (ZigBee PHY) and Bluetooth 5 PHY.

Definitions In *Concurrent Transmissions (CT)*, or *Synchronous Transmissions*, multiple nodes synchronously transmit the data they want to share. Nodes overhearing the concurrent transmissions receive one of them with high probability, due to the capture effect [87], or non-destructive interference. We shall note that we use both terms Concurrent Transmissions (CT), and Synchronous Transmissions interchangeably to mean tightly synchronized concurrent transmissions.



(a) Summing sinuous waves with different frequencies and phases results in a beating signal. Note that the two signals amplify and cancel each others periodically. See the sum of the signals around t = 0 – 0.1 and 0.3 – 0.4, for example.



(b) Capturing the envelope of a beating carrier in CT reception from two transmitters using two nodes and a software defined radio. The figure shows the envelope of the signal in the baseband after removing the carrier, and shall be a constant line in the case of the optimal transmitter.

Figure 1.2: Concurrent transmissions lead to a beating radio signal instead of having a uniform magnitude. This is due to the frequency offset of the commercial transmitters from the nominal standard frequency. Therefore, CT might become destructive if the signal distortion is severe.

Capture effect: A receiving radio can capture one of the many colliding packets under specific conditions related to the used technology [84, 87].

Non-destructive interference: If the colliding packets are tightly synchronized and have the same contents, then it is highly probable that they do not destruct each other; thus, enabling the receiving radio to recover the contents with a high probability. Ferrari et al. [42] presents an in-depth evaluation of this effect on 802.15.4, but they incorrectly assume it is *constructive interference*. Later work [91, 149] has shown that is not constructive in practice, but not totally destructive either; *i.e.*, the receiver decodes the packet with a high probability, but the concurrent transmission link quality is lower than the best single-transmission link. We confirm this as well when studying CT over Bluetooth [5].

Factors Affecting the Performance of CT In summary, the performance and practical feasibility of CT depend on four factors [149]: (i) the time delta between the two packets, and (ii) the Received Signal Strength (RSS) delta. Moreover, both (iii) the choices of the radio technology (modulation and encod-

ing), and (iv) whether the concurrently transmitted packets have an identical payload or not determine the range of the first two parameters for successful reception and the final robustness of the CT link.

In practice, the carrier frequencies of the different transmitters are never exactly equal. As a result, the concurrent transmission of the *same data* leads to a *beating* radio signal, where the signal magnitude alternates between peaks and valleys instead of being uniform, as illustrated in Figure 1.2. These variations in frequency and phase distort the signal; thus, CT might become destructive if the signal distortion is severe. It shall be noted that the radios transmit preamble bytes to synchronize the frequency and phase of the receiver to that of the transmitter. In the case of CT, the receiver would synchronize to the effective sum of the different preambles. On the other hand, the concurrent transmission of *different data* causes destructive interference of the signal that is only recoverable when one transmitter signal has an RSS delta sufficiently higher than the sum of the other CT as long as they are received within the duration of the signal preamble. 802.15.4 radios in the 2.4 GHz band utilize DSSS, where bits are encoded redundantly into *chips* with a 1:8 FEC redundancy, *i.e.*, 2 M chips/sec encode a 250 Kbps data stream, as highlighted earlier. This encoding helps to recover bits from the distorted signal in both cases of CT of the same and different data. Typically, in 802.15.4, the radio receives the stronger one of the concurrent transmissions if its signal is 3 dBm stronger, the so-called *co-channel rejection*, if they are synchronized within the preamble of 5 bytes, *i.e.*, 160 μs [84]. However, in the case of CT of the same data over 802.15.4, if the nodes transmit within 0.5 μs, then no signal strength delta is necessary [42]. On the other hand, radio standards that lack FEC mechanisms experience challenges when it comes to receiving CT [91].

Glossy is a flooding protocol for network-wide synchronization and data dissemination [42]. It established the design principle of concurrent transmissions of the same data in low-power wireless networks that are based on the IEEE 802.15.4 standard as it proved to be a highly reliable and efficient protocol. Glossy operates in rounds, with a designated node, the initiator, that starts the concurrent flooding. Nodes hearing the transmission synchronize to the network and join the flooding wave by repeating the packet. The transmissions are tightly synchronized to achieve non-destructive CT. Every node alternates between reception and transmission and repeats this multiple times to spread the information and achieve one-to-many data dissemination from the initiator to the rest of the network.

Chaos is an all-to-all data sharing primitive for low-power wireless networks [84]. Unlike current approaches, Chaos essentially parallelizes collection,

processing, and dissemination inside the network by building on two main mechanisms: Concurrent transmission and user-defined merge operators.

In Chaos, nodes synchronously send the data they want to share. Nodes overhearing these transmissions receive packets with a high probability due to the capture effect [87]. Upon reception, nodes merge the received data with their own and transmit the results again synchronously. Merging of data happens according to a user-defined merge operator. Chaos allows users to freely program various merge operators, from simple aggregates to complex computations. For example, Chaos computes simple aggregates, such as the maximum, in a 100-node multihop network within less than 90 milliseconds [84]. The whole process is triggered by an appointed node, the initiator, but continues in a fully distributed manner until all nodes in the network share the same data.

1.3 Related Work

We review related work categorized in the following aspects: (i) conventional networking protocols including (a) synchronized and scheduled MAC and (b) asynchronous, low-power routing; (ii) low-power channel hopping; and, (iii) concurrent transmissions.

1.3.1 Conventional Networking Protocols

This section reviews the work related to both scheduled MAC protocols and asynchronous low-power routing.

Synchronized and Scheduled MAC A synchronized and scheduled MAC protocol uses time division multiple access (TDMA) to manage medium access: It divides time into slots, synchronizes nodes to a common time-reference and schedules communications. The idea of synchronizing nodes and channel hopping to combat multipath fading and external interference is common in many technologies, including Bluetooth and cellular systems. It was brought to low-power wireless networking through a proprietary protocol called Time Synchronized Mesh Protocol (TSMP). Early promising results [29] resulted in standardizing the core technology of TSMP as WirelessHART [44], ISA100.11a [63] and IEEE802.15.4e [61].

In a WirelessHART network, a central entity computes the communication schedule based on application requirements and on information it gathers about network connectivity. The schedule is injected into the network and continuously updated throughout the lifetime of the network [74]. In static networks with predictable traffic patterns, commercial TSCH-like networks

such as WirelessHART or SmartMesh IP [146] offer very high reliability (published results include 99.999% on a 49-node industrial deployment [29], 99.95% on a 15-node testbed [113]), and a decade of battery lifetime [148]. Note that the centralized scheduling algorithms are not part of the standards and much attention has been given to scheduling theory in the context of TSCH networks [110, 124, 125, 158].

In scenarios where the network topology and traffic patterns are not fixed, decentralized scheduling is applicable. Tinka et al. [140] present, to the best of our knowledge, the first paper to propose and demonstrate a decentralized scheduling solution for TSCH networks. A common shared slot is used for neighbor discovery and for negotiating the addition of dedicated slots. This work motivated further approaches [101, 109, 157]. For example, Morell et al. [101] take schedule negotiation one step further and propose a multihop reservation through label switching.

Focusing on very low data rates and not limiting itself to TSCH, Dozer [17] employs distributed scheduling for energy-efficient routing. Its goal of high energy efficiency leads to significantly high latency. For example, experiments report an average latency of 30 s and a PDR of 98.5% at a radio duty-cycle of 0.2% for a testbed of 90 nodes [40].

In October 2013, the IETF created the 6TiSCH working group, which standardizes how to use an IPv6-enabled upper stack on top of IEEE802.15.4-TSCH. 6TiSCH specifies a number of mechanisms to manage the TSCH schedule [139]. It defines a CoAP-based management protocol which can be used for central scheduling, and a protocol for neighbor nodes to negotiate distributed scheduling. 6TiSCH, however, leaves it to the implementer to decide which scheduling approaches fit best. Finally, Deguglielmo *et al.* [58] proposed an analytical model of the back-off mechanism in TSCH, taking into account the occurrence of capture effect in shared slots.

Asynchronous, Low-power MAC and Routing At the other end of the spectrum, an asynchronous protocol manages medium access by carrier sense (CSMA): It senses the medium, delays transmissions until it believes it is free from interference or competing transmissions, backs-off on collisions and retransmits. When used together with low-power routing, it enables dynamic applications and transparently allows nodes to join and leave, without planning or scheduling. However, its packet loss can be up to several percents: For example, CTP reports delivery ratios between 94% and 99.9% in data collection depending on the network size and topology [51]. Recent approaches such as ORW [85], ORPL [36] and BFC [114] improve in terms of energy efficiency and – in part – latency over CTP but still have an average PDR of roughly 99%. Moreover, these do not employ scheduling and channel hopping;

thus, they cannot avoid external interference, multi-path fading, or contention efficiently. EM-MAC [138] and MiCMAC [4] integrate channel hopping into asynchronous, low-power routing. As a result, they increase reliability, especially in the presence of external interference, but channel hopping with the loosely synchronized operation leads to an extra overhead in terms of latency and radio duty-cycle [4]. It shall be noted that similar techniques were standardized later in the CSL and RIT MAC modes of IEEE 802.15.4-2015 [61].

1.3.2 Low-power Channel Hopping

Using frequency diversity techniques has proven to be effective for combating interference [145]. It is wide-spread both in established standards, such as Bluetooth [12], TSCH [61] and in the state of the art such as the top solutions in the dependability competition [129]. By default, TSCH uses all available 16 channels of the IEEE 802.15.4 PHY layer. Hopping over all available channels enables reliable operation even when a subset of the channels has poor quality. Adaptive channel blacklisting for TSCH pushes reliability and throughput further [39]. BLEach [135], implements IPv6 over BLE as in RFC 7668 [69]. Moreover, it enables adaptive channel blacklisting and adaptive duty cycling to provide quality of service guarantees. However, it only supports star networks but not multihop Bluetooth mesh networks.

1.3.3 Synchronous Concurrent Transmissions

In this section, we discuss the state of the art in the broader field of concurrent transmissions and related phenomena such as constructive interference and capture effect. We also discuss the protocols that base on these concepts in Wireless Sensor Networks (WSNs) and Internet of Things (IoT).

Understanding Concurrent Transmissions While the capture effect is not new and was first observed for FM transmitters [87], the capture effect in low-power wireless networking was first experimentally studied by Ringwald and Römer [121] over On-Off-Keying (OOK) modulation where they design BitMAC, a MAC protocol that utilizes CT to implement simple in-network aggregates to provide collision-free communication. Later, Son *et al.* [134] experimentally studies CT over 802.15.4 compatible radios. The success of concurrent transmissions in Glossy started a debate on how CT works and what underlying physical phenomena enable it. The authors of Glossy argue that the signals interfere constructively. Later, this was underlined by Rao *et al.* [119] who utilize Glossy style flooding and through precise timing can also achieve destructive interference to provide negative feedback.

In contrast, Wilhelm *et al.* [149] introduce analytical models backed with experiments to parameterize concurrent transmissions and show that these are rather non-destructive interference instead. Thus, they argue that the signals get degraded due to concurrent transmissions but still can be decoded. Moreover, they argue that coding is essential to improve the reliability of concurrent transmissions. Similarly, Liao *et al.* [91] argue that DSSS and its coding is what lets CT survive beating. While the mentioned papers are limited to 802.15.4 in the 2.4 GHz band, Liao *et al.* [92] have a limited study on CT over 802.15.4 in sub-GHz. Roest [122] studies the capture effect and evaluates Chaos on BLE, 1 Mbps. To the best of our knowledge, no prior research has evaluated and utilized CT over Bluetooth 5 extensively.

Concurrent Transmissions Protocols BitMAC [121], A-MAC [38] and Glossy [42] pioneered the field of concurrent transmissions in WSNs. Glossy provides network-wide flooding, *i.e.*, from a single sender to all nodes in the network, on a millisecond scale. Others, such as LWB [40], Splash [28], Choco [136] base on Glossy to schedule individual network-floods to provide data collection. Crystal [65] and its multichannel version [66] reduce the number of Glossy floods by relying on data prediction. CXFS [18], Sparkle [156] and others [16, 68, 70, 127, 155] limit the number of concurrent transmitters in Glossy or LWB while Sleeping Beauty [126] combines both limiting the number of transmitters by putting them to sleep and scheduling Glossy floods to improve energy efficiency. Baloo [71] provides a framework for easing the development and implementation of Glossy-based synchronous transmissions protocols.

SurePoint [9] builds an efficient concurrent network-wide flooding similar to Glossy in UWB and leverages it to provide a localization service while Corbalán and Picco [25] introduce concurrent ranging on UWB. In concurrent ranging, a transceiver tag estimates the distance to anchors by exploiting the channel impulse response (CIR) estimation of the anchors' replies. The CIR estimation function is available on standard UWB transceivers. SanpLoc [56] takes this further by cleverly assigning a fixed delay for every anchor response, allowing the tags to localize by listening to the simultaneous back-to-back responses of anchors, without the need of previous knowledge of anchors locations.

Chaos [84] on the other hand extends the design of Glossy to utilize the capture effect on 802.15.4 in the 2.4 GHz band to let nodes transmit different data and efficiently calculate network-wide aggregates by employing in-network data processing. Mixer [60] and Codecast [99] utilize network coding techniques for efficient many-to-many data sharing.

However, since these approaches base on the capture of different data rather than flooding the same data, it is more difficult to support them on uncoded communication technologies such as the Bluetooth modes 1 and 2 Mbps.

1.3.4 Network-wide Agreement and Transactions

Achieving agreement in conventional distributed systems is a mature research field, with solutions such as 2PC [53] and 3PC [133], or PBFT [19] for the even more challenging context of Byzantine Fault Tolerance. Paxos is a general (non-Byzantine) fault-tolerant solution to the consensus problem [81, 82]. Paxos has been extended and further optimized, for example with Fast Paxos [80], Cheap Paxos [83] or Ring Paxos [95]. Raft [107] is an alternative to Paxos, designed to be more comprehensive and easier to understand and implement as such protocols are very complex in nature and implementations are error-prone. Wireless sensor networks, however, bring unique challenges, in particular, the low-power, multihop nature of the network, and the lossy links.

Existing literature covers distributed data processing and aggregation [72] in WSN, but more demanding primitives such as atomic transactions, fault-tolerant consensus, and reliable multicast are mostly limited to simulation or modeling [59, 78, 93, 94]. While consensus has been studied in opportunistic and ad-hoc networks [11, 24], practical approaches are limited to single-hop networking [13, 27, 115] or have a latency of several seconds [41]. On one hand, for example, JAG [13] provides a reliable agreement between pairs of neighbors with a focus on interference resilience but does not address network-wide agreement, and Moniz *et al.* [100] studied consensus with Byzantine failures in single-hop networks. The 6P protocol [115] is being designed as part of IETF 6TiSCH for schedule negotiation; it offers transaction between pairs of neighbors based on a 2/3-way handshake. On the other hand, Köpke [77] proposes an adapted 2PC for WSNs, while Borran *et al.* [15] extends Paxos with a new communication layer for opportunistic networks over the MAC layer of 802.11. Both solutions build a tree to collect and route responses and suffer from the overhead of maintaining the tree and collecting responses with unicast.

1.3.5 Summary

Conventional asynchronous protocols or even TDMA approaches such as TSCH [35, 61] when combined with routing protocols such as RPL [137], CTP [50] provide best-effort low-power routing. WSN rate-controlled protocols [30, 67, 75, 108, 117] or TCP for 6LoWPAN [37] enable end-to-end reliability on top of a best-effort routing protocol. While TDMA-style protocols have high reliability, these protocols suffer from significantly higher latencies when

adapting to the changing wireless environment as they need to recompute and distribute schedules. Moreover, it is hard to support mobility in scheduled protocols, as opposed to the approaches based on synchronous transmissions [40, 60, 84].

Further, supporting group communication on top of a routing protocol is a challenge *per se*. Solutions such as BMRF [45] or SMRF [106] provide such feature in a RPL multicast context, but they exhibit high latencies and loss rates of a few percents or tens of percents. Most importantly, existing group communication protocols are best-effort, *i.e.*, they do not provide end-to-end reliability. The lack of end-to-end acknowledgments combined with the often significantly higher latency makes it very challenging to realize advanced communication primitives, such as group membership and network-wide consensus.

However, synchronous transmissions always need one central entity to control and initiate the synchronous transmissions. Moreover, they – inherently – have a high channel utilization and are often limited in terms of generality such as being restricted to periodic traffic. Therefore, conventional routed protocols are better suited to general-purpose applications with RPL and IPv6.

1.4 Thesis Contributions and Roadmap

In this section, we first set the context of our work, identify concrete goals and informally describe the journey we take to achieve these goals. Then, we dive deeper with introductory summaries of the thesis contributions that give the reader a deeper, yet concise, view of the rest of the thesis.

1.4.1 Thesis Roadmap and Goals

Based on our problem statement §1.1.1, the requirements we introduced in §1.1.3, and the review of the related work §1.3, we list the following concrete goals and we informally describe our journey in tackling them.

A. Goal: Standard-Compliant Highly Reliable Communication

A class of applications in the Internet of Things, such as building monitoring and automation systems, requires highly reliable delivery of events that do not have a specific pattern or a pre-defined schedule. At the same time, end-to-end Internet connectivity and standard compliance are demanded for easy interfacing with other systems, for example, smart lights and indoor environmental control.

We build on TSCH as synchronized and scheduled protocols have demonstrated high reliability and low-energy. On one hand, we want to operate TSCH

without the need of a centralized scheduler, as it takes time to react to network conditions, and until the new schedule arrives, the network might change. Thus, we will be tracking a continuously changing state, unless the network is perfectly isolated from external interference and moving objects inside, which is an unrealistic demand. On the other hand, decentralized schedulers need complex algorithms and have a signaling overhead which might make them impractical. Moreover, to support end-to-end traffic, the network needs routing that integrates with the communication schedules, such that the routes and schedules stay consistent as well. This dictates the following options: Either the scheduler provides routing tables, or we need another protocol to maintain routes. We note that routing is widely studied, and RPL is the standard state of the art routing protocol for low power and lossy networks.

Approach *Support RPL over TSCH and infer schedules autonomously based on the routing topology.*
We summarize our work, Orchestra, that is tackling this issue in §1.4.2.A.

B. Goal: Network-Wide Consensus

Many applications build their operations on consensus; however, achieving consensus requires to: (i) Disseminate a proposal, (ii) collect responses, (iii) disseminate the decision, and optionally (iv) collect feedback. Realizing a consensus service requires several one-to-many and many-to-one communication steps. Thus, it is more suited to many-to-many communication styles, commonly supported by concurrent transmission protocols like Chaos [84] and LWB [40]. On the contrary, realizing these steps using conventional unicast- and routing-based protocols is slow and relatively expensive, and implementing any of these steps, *e.g.*, collecting responses, would take a latency of seconds in state of the art conventional protocols [4, 34, 37, 102]. We conjecture that achieving a network-wide consensus would take several seconds using conventional methods. This is prohibitively slow for applications that require rapid consensus, *e.g.*, coordinating drones' maneuvers.

Orchestra *c.f.*, §1.4.2.A. is no exception, as it depends on routing. It is mostly suited to converge-cast traffic and suffers from high latencies in the range of seconds to minute. Moreover, we notice that the capture-effect contributes significantly to the increased reliability of the shared links in Orchestra, so we turn to investigate concurrent transmissions with in-network processing to speed up consensus. Therefore, we liberate us from standard compliance and adopt the concurrent transmissions style with in-network processing. We conjecture that realizing rapid consensus services requires a rethinking of the design of the conventional protocols and is more suited to concurrent transmissions based designs.

Approach *We inspire from TSCH [62] and Chaos [84] and design a time-slotted concurrent transmissions MAC layer. We realize network-wide consensus, group membership, collection and aggregation services using two-phase commit and in-network processing.*
We summarize our work, A^2, that is tackling this issue in §1.4.2.B.

C. Goal: Fault-Tolerant Consensus

Fault tolerance is a necessity in most practical systems and a must in mission-critical systems. A regular user would like his system to continue working despite a failure in one component, and such system failures could be catastrophic or incur financial losses in mission-critical systems. Besides, supporting fault tolerance is of utmost importance in systems that build their operations on consensus.

A^2 supports network-wide consensus based on two-phase commit (2PC). However, 2PC suffers from blocking when failures happen. For that, we turn to Paxos, which is fault-tolerant by design but has a relatively high messaging overhead.

Approach *We adapt Paxos to the in-network processing and many-to-many communication scheme supported in A^2 and present Wireless Paxos.*
We summarize our work, *Wireless Paxos*, that is tackling this issue in §1.4.2.C.

D. Goal: Enable Efficient Multihop Connectivity over Bluetooth 5

Bluetooth has a wide install base and a high adoption rate as it is available on most modern phones and laptops. Moreover, the recent Bluetooth 5 presents four operating modes that offer high data-rates of 1 and 2 Mbps and slower but more reliable modes that use forward error correction codes (FEC) with 500 and 125 Kbps. However, the focus of Bluetooth has been on star networks, but recently, Bluetooth mesh was introduced, which is based on flooding. State of the art in Wireless Sensor Networks shows that concurrent transmission approaches, *e.g.*, Glossy [42], are superior to other alternatives, especially, when it comes to supporting fast and reliable network-wide flooding. Nevertheless, most of the research has been focused on IEEE 802.15.4. Since Bluetooth has a physical layer different from 802.15.4, the applicability of concurrent transmissions on top is questionable.

Approach *We study the feasibility of concurrent transmissions over Bluetooth 5 and design a protocol for multihop Bluetooth networks.*
We summarize our work, *BlueFlood*, that is tackling this issue in §1.4.2.D.

1.4.2 Thesis Contributions

This section summarizes the papers that constitute the main body of the thesis. We employ experimental computer science methodology in our research: We design and implement protocols targeting real-world systems, and we evaluate in both simulations and testbeds.

A. Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH

Context and Challenge With the emergence of the Internet of Things (IoT), there is a need for a network protocol that is flexible, supports non-deterministic applications and at the same time is highly reliable. Example applications range from smart homes, *e.g.*, smart lighting, to smart cities, *e.g.*, smart parking lots, including wearable consumer devices. state of the art solutions have loss rates in the range of one percent [36, 51, 76, 85].

Such a loss rate renders these applications useless, buggy or at best cumbersome to use. For example, in a smart lighting scenario, think about one light not turning on, or in a paid parking garage, one car being charged though it left, as the system failed to sign it off due to a lost message. No developer would employ such a network protocol without end-to-end reliability, as done in TCP, for example. However, with end-to-end reliability, losses trigger costly re-transmissions which often come in a burst and result in a jittery performance and wasted energy. At the other end of the spectrum, robust mesh networking solutions that are designed for applications with explicitly scheduled traffic can achieve 2 or 3 orders of magnitude fewer losses, *i.e.*, up to one loss per 10.000 packets [17, 29, 40, 113, 145]. Often, these are tailored to deterministic applications with periodic traffic and have strong assumptions on the network topology and its wireless link dynamics. We investigate how to apply such highly reliable scheduled solutions in non-deterministic scenarios in the context of IoT.

Orchestra in a Nutshell We make a case for autonomous TSCH (Time Slotted Channel Hopping [61]) scheduling in non-deterministic low-power RPL and IPv6 networks. The key challenge we address is that of creating TSCH schedules without hindering any of the flexibility of RPL networks and matching the requirements of non-deterministic applications. We introduce Orchestra, a new approach to scheduling in any scheduled MAC layer and any routing protocol. The focus of this chapter is on the TSCH MAC layer and the RPL routing protocol.

Traditional TSCH scheduling solutions such as WirelessHART [44] and ISA100.11a [63] rely on a centralized scheduling entity, and the new standards

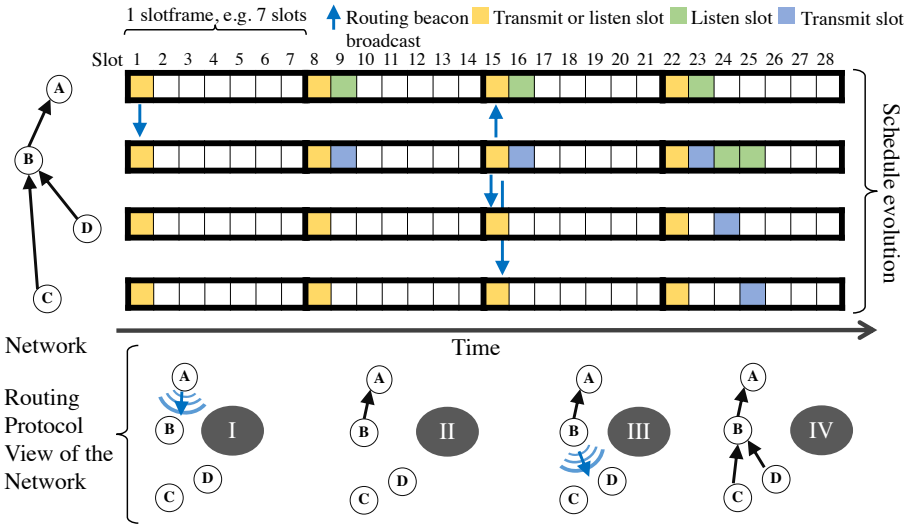


Figure 1.3: Orchestra manages nodes schedules according to rules and to the evolving network topology. This example demonstrates a converge-cast (or data collection) example. We use two overlapping slotframes of 7 slots per each node that repeat periodically, one for routing beacons and one for data collection, but we draw them as one here. The first slot is reserved for beaconing transmission and reception. As the routing protocol broadcasts beacons and the directed routing tree is built, each parent node reserves reception slots to listen to each of its children, and each of the children reserves a slot to transmit to its parent.

being developed in the IETF 6TiSCH working group [139] employ schedule negotiation between neighbor nodes. Orchestra is radically different from existing scheduling solutions in that it does not involve any extra central entity, negotiation, signaling, nor multihop path reservation among nodes. Instead, nodes employ simple periodic schedules and update the schedules automatically and instantly as the routing topology evolves.

A TSCH schedule in Orchestra consists of a set of over-provisioned communication slots dedicated each to a specific communication plane: MAC, routing, and application. A developer defines a set of *scheduling rules* that stipulates the schedules based on the topology information, the traffic plane and a defined period. As a result, Orchestra allows building a generic, flexible, low-power routing backbone using RPL while benefiting from the robustness of TSCH.

An Orchestra schedule contains different slotframes of different lengths. Each slotframe is dedicated to a particular type of traffic: TSCH beacons, RPL

signaling traffic or application data. Nodes select slots using the scheduling rules which reduce contention drastically or in certain cases eliminate contention. This makes Orchestra particularly appealing for low-power IPv6 scenarios where different applications generate event-based data, without any pre-defined (*e.g.*, periodic) traffic pattern.

The following is a concrete example of a set of Orchestra *scheduling rules*:

- A dedicated broadcast slot from every node to its children for TSCH beacons, repeating every X slots;
- A slot common for all nodes in the network for either broadcast or unicast for RPL signaling, repeating every Y slots;
- A dedicated unicast slot from every node to its RPL preferred parent, repeating every Z' slots;
- N dedicated unicast slots from every node to each of its children, repeating every Z" slots.

The key is that we select the time and channel offset of every slot as a function of the sender's or the receiver's identifier (MAC address or a unique network node ID). Based on these rules, Orchestra builds and evolves the schedules as the routing topology evolves. For example, the first rule stipulates a slot from every node to its children. Consequently, when a child node changes its parent, it will automatically remove the slot associated with the old parent and replace it with a slot associated with the new. Moreover, Orchestra can either attain very low levels of contention or operate contention-free, depending on the scheduling rules. For example, to guarantee a contention-free operation, a scheduling rule may stipulate a slotframe's length equal to the number of nodes, and dedicate slots to nodes based on their IDs. Figure 1.3 illustrates Orchestra in a data collection scenario.

Results We implement Orchestra and TSCH in Contiki [33], and experiment in two testbeds with 98 and 25 nodes, each with a different hardware platform. In total, our evaluation bases on 219 individual experiments, up to 72 hours long, and a total of 1,178,601 UDP packets routed from source to destination. We show that Orchestra enables autonomous TSCH scheduling in RPL networks and limits the loss rates to 10^{-4}, end-to-end. This is an improvement of 1 to 2 orders of magnitude over state of the art asynchronous solutions such as RPL with ContikiMAC. We show that Orchestra achieves this strong reliability while keeping energy and latency close to the state of the art.

Contributions This chapter makes the following contributions:

- The main contribution of this chapter is Orchestra, a system that allows TSCH nodes to maintain their schedules autonomously, driven by the state of the routing protocol. Orchestra operates without a centralized scheduler, and inter-node schedule negotiation nor path reservation;

– We demonstrate experimentally that Orchestra is practical, scalable, and achieves end-to-end loss rates two orders of magnitude below asynchronous low-power listening.

Besides, this chapter presents findings on how TSCH compares to low-power listening, independent of Orchestra:

– Even without channel hopping, the time-slotted nature of TSCH improves network connectivity and reduces medium utilization;
– Although it requires global synchronization, TSCH is practical in sparse traffic scenarios, and helps achieve high reliability in networks running a distributed routing protocol such as RPL [137], and
– TSCH can keep nodes tightly synchronized, enough to enable the radio capture-effect in shared slots.

TSCH and Orchestra were contributed to the open-source project ContikiOS and later to Contiki-ng.

Statement of Personal Contribution I am a co-designer of Orchestra and a main implementer of the TSCH MAC layer. I implemented the static scheduler and co-designed and conducted the experimental evaluation comparing it to Orchestra and participated in the write-up of the manuscript.

This chapter was published as a paper in the Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys), 2015 [34].

B. A^2– Agreement in the Air: Network-wide Consensus Utilizing the Capture Effect in Low-power Wireless Networks

Context and Challenge Many applications in low-power wireless networks build their operation on consensus: For example, networked cooperative robots and UAVs agree on maneuvers to execute [7]; wireless closed-loop control applications such as adaptive tunnel lighting [21] or industrial plants [103, 105] agree on set-points for actuators. Within the network stack, protocols need to agree on which cryptographic keys to use [31], which channel hopping and transmission schedules to follow [61], or which nodes to elect as cluster heads [152].

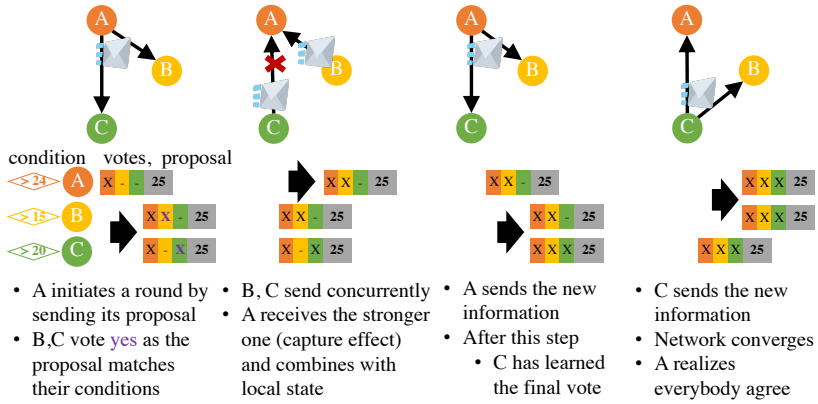
These application scenarios exhibit key differences when compared to traditional data collection or dissemination in wireless sensor networks: They demand primitives for network-wide consensus at low latency and highly reliable data delivery with robustness to interference and channel dynamics [3]. For example, after rolling out new cryptographic keys or channel hopping schemes in a network, the new configuration can only be applied once a network-wide agreement has been reached that all nodes are aware of the new data. Otherwise, nodes might be excluded and would need to re-join the network. This requires multi-phase agreement protocols such as two-phase commit [53]. Protocols for distributed consensus are mature solutions in a wired context, such

as for data centers or databases, but have received little attention in low-power wireless settings.

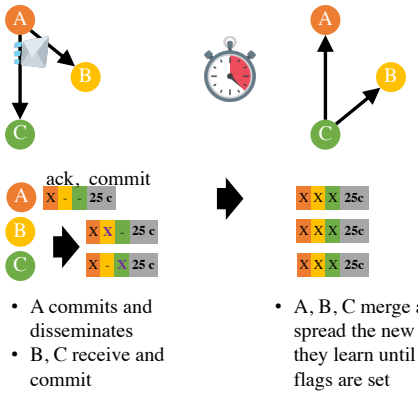
Approach We divide the operation of the applications that build on consensus into two steps: (i) Decision making and agreement, and (ii) executing the agreed-on action. For example, in the networked cooperative robots' and UAVs' agreement on maneuvers, the nodes need to first agree on the action to take, then they execute that action. The execution of the action, *e.g.*, taking a specific maneuver, needs specialized algorithms like closed-loop control methods, which are out of the scope of this thesis. We focus on decision making and agreement, rather than how to execute the agreed-on action. We utilize the distributed systems' consensus methods for agreement, which are orthogonal to the information consensus methods popular in the electrical engineering domain, as in [120] for example.

We argue that the low latency of new approaches to synchronous transmissions, such as Glossy [42] and Chaos [84], are key enablers for distributed consensus protocols in low-power wireless networks. We introduce A^2: Agreement in the Air, which builds on a new synchronous transmission kernel, Synchrotron. Synchrotron extends the concepts introduced by Chaos with high-precision synchronization through VHT [128], time-slotted operation, a network-wide scheduler, frequency hopping and multiple parallel channels, and security features (in part inspired by LWB [40] and TSCH [35, 61]). On top of this robust base layer, we tackle the consensus challenge and show how to enable two- and three-phase commit protocols (2PC and 3PC) [53, 133] in low-power wireless settings. In addition, we address the consistent group membership problem and build reliable primitives for nodes to join and leave the network. Applications can use A^2 to reliably agree on, for example, cryptographic keys, channel-hopping sequences or set-points for actuators, even in the presence of node or link failures. Next, we illustrate A^2 concepts by discussing how it realizes the two-phase commit primitive.

Two-phase Commit in A^2 We achieve 2PC with two phases of synchronous transmissions back-to-back within a single round. First, in the voting phase, the coordinator proposes a value and collects the cohort's votes. Once the result of the voting phase reaches the coordinator, it decides to commit if and only if all nodes accepted the proposal; else, *i.e.*, after a timeout or after receiving one or more votes against its proposal, it aborts. Second, in the commit or abort phase, the coordinator disseminates the outcome of the vote, *i.e.*, commit if all agree, or abort otherwise. Upon reception, nodes switch to the new phase, set their flags, adopt the final result and continue the dissemination. This way, the two phases of voting and commit/abort are interleaved rather than strictly segregated, for efficiency. This is illustrated in Figure 1.4.



(a) Two-phase commit: Collect votes.



(b) Two-phase commit: Commit on value.

Figure 1.4: Two-phase commit in A^2: Node A proposes value 25. Nodes B, C agree by voting for it. This result propagates back to the coordinator, who initiates a commit phase. At the end of the second phase, all nodes have reached the consensus to commit. Note how votes collection, processing, and acknowledgment is done with in-network processing.

Results We implement and evaluate Synchrotron and A^2 on four testbeds ranging from 29 to 213 nodes. In our experiments, A^2 completes a 2PC round over 180 nodes within 475 ms at a 0.5% duty-cycle for 1-minute intervals. 3PC completes within 900 ms at a 0.85% duty-cycle under similar settings. Synchrotron reliably aggregates the maximum value periodically from over 200 nodes with *zero losses* end-to-end, over an extensive experiment collecting millions of data points.

Moreover, we evaluate 2PC and 3PC's liveness and consistency under emulated failures. We show that our implementation behaves as expected: On one hand, we notice that *2PC* delivers on the promise of eliminating the inconsistencies, as it orders the nodes that are in the uncertainty state to block (*i.e.*, those that voted yes and failed to hear the outcome). The price to pay is liveness, as the whole protocol has to block until all nodes are available again for recovery. On the other hand, *3PC* results in few inconsistencies in favor of liveness. We conclude that for applications that cannot afford inconsistencies, 2PC is a safe solution. For applications that cannot afford to block but can do with little inconsistencies, 3PC is a practical option. Otherwise, a fault-tolerant consensus protocol, like Paxos [82], shall be considered; *c.f.*, §1.4.2.C.

Contributions This chapter makes the following contributions:

- We introduce network-wide voting, based on synchronous transmissions;
- We build network-wide consensus protocols for low-power wireless: two- and three-phase commit;
- We devise a consistent group membership protocol based on network-wide transactions; and
- We present Synchrotron, A^2's underlying kernel for synchronous transmissions, which provides distributed schedules, the ability to utilize multiple frequencies in parallel, and authentication and encryption to ensure robust and fast agreement.

A^2 and Synchrotron are available as open source[1].

Statement of Personal Contribution I am the lead designer and implementer of A^2 and Synchrotron. In addition, I designed and conducted the experimental evaluation and wrote a major part of the manuscript.

This chapter was published as a paper in the Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys), 2017 [6].

C. Paxos Made Wireless: Consensus in the Air

Context and Challenge Many applications in low-power wireless networks need a solution for fault-tolerant consensus, since such low-power wireless networks are prone to faults, nodes running out of battery, message losses, and network segmentation. However, failures complicate the process of reaching an agreement. It is even proven impossible to achieve consensus when at least one node is never able to communicate [43].

In the distributed systems community, many solutions to the consensus problem have been proposed [82, 97, 107]. Paxos is one of the first protocols to provide fault-tolerant consensus [81, 82] in a non-Byzantine setting: Paxos will

[1] `https://github.com/iot-chalmers/a2-synchrotron`

lead to a correct consensus as long as a majority of nodes are participating and all nodes will eventually learn the correct outcome as long as the majority accepted the decision. Paxos is often used in an extended and optimized form, Multi-Paxos [82], which allows nodes to agree on a continuous stream of values and enables state machine replication. For example, UAVs can continuously coordinate their next destination with Multi-Paxos.

Paxos and Multi-Paxos have become the default protocols to ensure consistent data replication within data-centers, *e.g.*, Google's Chubby locking mechanism [23], Microsoft's data-center management Autopilot [64], and IBM's data-store Spinnaker [118].

However, the complexity of Paxos, and its many required interactions pose key challenges in low-power wireless networks. Devices in WSNs have strong resource constraints in terms of bandwidth, energy, and memory. Radios are, for example, commonly duty-cycled to save energy [32, 102]. In contrast, Paxos requires many message exchanges and high bandwidth to reach consensus.

Additionally, Paxos has been initially designed for wired networks, and is, therefore, heavily influenced by their unicast structures. Later work shows that Paxos overhead can be optimized by utilizing multicast [15], or by introducing an additional logical ring to reduce communication overload [95]. However, these approaches still rely on unicast for parts of the algorithm.

In contrast, low-power wireless networks are broadcast-oriented networks where each transmission can be received by all neighboring nodes. Executing unicast-based schemes in wireless networks usually induces higher costs, especially in multihop networks. Moreover, multihop networks also provide opportunities for data aggregation and computation of intermediate results, which are not part of Paxos' design rationale.

Approach In this chapter, we bring fault-tolerant consensus to low-power wireless networks. We propose WPaxos (*Wireless Paxos*), a new variant of Paxos fitted to the characteristics of low-power wireless networking:

We *co-design* Paxos with the lower layers of the network stack to provide network-wide consensus at low latency. Specifically, we base the design of WPaxos on three principles:

- *Broadcast-Oriented Communication:* We take advantage of the broadcast properties of the wireless medium to disseminate requests and collect responses efficiently from all nodes.
- *Concurrent Transmissions:* Like A^2, we build on top of concurrent transmissions to provide low latency and high reliability.
- *Local Computing and Aggregation:* We distribute the decision logic to all nodes through an aggregation function to convert Paxos to a many-to-many scheme.

Next, we design, WMulti-Paxos, a primitive that provides the functionality of Multi-Paxos for agreement on a continuous stream of values and state machine replication. Multi-Paxos allows multiple Paxos rounds to be executed at once, by aggregating all the requests into one message, which results in a lower duty cycle and lower latency. Our design also allows agreeing on multiple values at once by piggybacking consecutive requests in order to reduce the amount of data transmitted.

The overall result is a broadcast-driven consensus primitive using in-network processing to compute intermediate results in Paxos. Our solution builds on top of A^2 and Synchrotron [6] that provide a basis for highly reliable and low-latency networking in low-power wireless with support for in-network processing.

Results We implement and evaluate WPaxos and WMulti-Paxos on two testbeds — Flocklab and Euratech, of 27 and 188 nodes, respectively. In Euratech, WPaxos, and WMulti-Paxos take 289 ms and 133 ms for the agreement among the majority, and it takes 633 ms and 500 ms for network-wide dissemination of the agreement result. It shall be noted that Paxos considers a consensus complete once a majority accepts. For comparison, 2PC has about 1.4× the latency of WMulti-Paxos, and 3PC takes 2.4×.

Moreover, we evaluate the consistency of WMulti-Paxos under injected failures. We show that it sustains a correct consensus without blocking, as opposed to 2PC that blocks to handle failures.

Contributions This chapter makes the following contributions:

- By distributing parts of the consensus logic, we show that Paxos can be expressed as a many-to-many communication scheme, rather than a multicast scheme;
- We present *Wireless Paxos*, a new variant of Paxos specifically designed to address the challenges of low-power wireless networks, and *Wireless Multi-Paxos*, an optimized extension of Wireless Paxos for continuous streams of agreed values for constrained devices;
- We implement and evaluate our contributions on two testbeds, composed of 27 and 188 nodes and compare our results to solutions from the literature.

WPaxos and WMulti-Paxos are available as open source[2].

Statement of Personal Contribution I have a minor role as a co-designer and co-implementer of Wireless Paxos. I extended the underlying protocol Synchrotron to improve the performance of Wireless Paxos, as well.

This chapter was published as a paper in the Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN),

[2] `https://www.github.com/iot-chalmers/wireless-paxos`

2019 [112]. The paper was nominated as a *candidate for the best paper award* at the conference.

D. BlueFlood: Concurrent Transmissions for Multi-Hop Bluetooth 5

Context and Challenge Bluetooth is omnipresent communication technology. In 2017, more than 3.6 Billion Bluetooth-enabled devices were sold and the overall installed base of Bluetooth devices is estimated to be roughly 10 Billion [1]. This makes Bluetooth predominant in our modern, connected society. In the past decade, the research community has designed a plethora of MAC, routing, and dissemination protocols for low-power wireless networking. However, the focus for networking in low-power wireless has been nearly exclusively on IEEE 802.15.4. For example, Glossy [42] made a breakthrough in low-power wireless in disseminating information at network scale quickly and efficiently. It utilizes concurrent transmissions of tightly synchronized packets to realize flooding and synchronization services. As of today, Glossy is practically limited to 802.15.4 in the 2.4 GHz band and – to a smaller degree – ultra-wideband communication (UWB) [9, 25] and 802.15.4 in the sub-GHz band [18].

Concurrent transmissions for Bluetooth, however, have been overlooked until today. It is, for example, not shown whether the concepts of concurrent transmissions apply to Bluetooth, as Bluetooth uses a completely different physical layer from the one used in IEEE 802.15.4 in the 2.4 GHz band *c.f.*, §1.2.2.

Approach We argue that adapting the concepts of concurrent transmissions to Bluetooth can open a variety of new application scenarios due to the ubiquitous availability of Bluetooth-enabled devices. In this chapter, we design and evaluate concurrent transmissions on top of Bluetooth PHY. Later, we exploit them in BlueFlood to provide network-wide flooding.

We present BlueFlood: A network stack based on concurrent transmissions to provide low power, low-latency, and reliable flooding and data dissemination to Bluetooth mesh networks that are battery operated.

Overview of BlueFlood BlueFlood utilizes CT of the same data, as depicted in Figure 1.5. We take inspiration from Glossy and A^2 [6] and design our protocol to be a round-based and time-slotted protocol. Thus, just like in Glossy, Chaos and A^2, we schedule individual communication rounds on a network-wide scale. In the beginning of a round, all nodes wake up aiming to receive. A round is further split into time slots in which nodes either transmit, listen or sleep, according to a so called transmission policy.

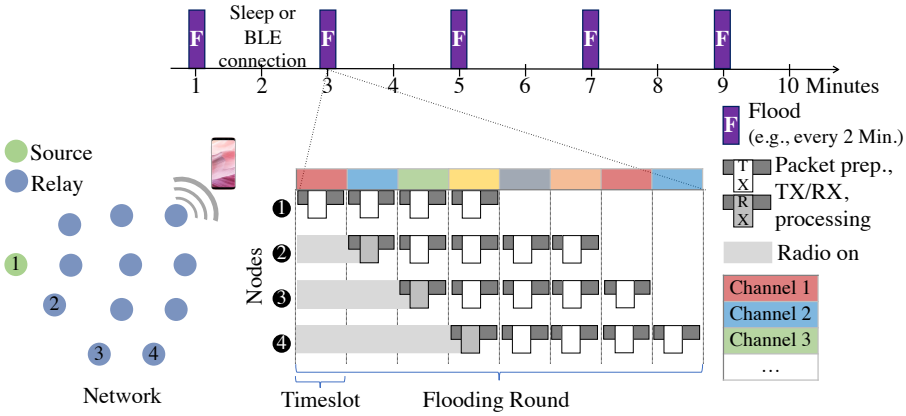


Figure 1.5: Overview of BlueFlood: It schedules rounds for network-wide dissemination that utilizes synchronous flooding. Each round has multiple slots in which nodes transmit, receive or sleep, driven by the transmission policy: This example uses RX / 4 TX. Each node hops the channel every timeslot. Between two rounds, nodes either sleep or are free to run other Bluetooth connections.

Results We evaluate BlueFlood in a residential environment and show that BlueFlood achieves 99% end-to-end delivery ratio in multihop networks with a duty cycle of 0.13% for 1-second intervals. Moreover, we show the fragility of CT over Bluetooth. We show that on one hand, the reception of concurrent transmissions of different data is fragile when the uncoded modes are used and only practical with the low bitrate coded mode 125 Kbps. On the other hand, the concurrent transmission of the same packet is practical, and we use it as the base for designing BlueFlood.

Contributions This chapter makes five key contributions:

- We demonstrate the practical feasibility of concurrent transmissions on Bluetooth PHY.
- We evaluate the performance trade-offs of the four different transmission modes provided by Bluetooth 5 of 1 and 2 Mbps and coded long-range modes with 500 and 125 Kbps, for concurrent transmissions.
- We introduce BlueFlood: a multihop, low-power concurrent flooding protocol for Bluetooth PHY.
- We demonstrate that BlueFlood is received by off-the-shelf receivers, *e.g.*, smartphones.
- We illustrate how modern System-On-Chip (SoC) hardware simplifies the design of protocols based on concurrent transmissions.

BlueFlood is available as open source[3]. This includes the code, the experimental data and the scripts needed to reproduce our results.

Statement of Personal Contribution I am the sole designer and implementer of BlueFlood. In addition, I designed and conducted the experimental evaluation. I am also the lead author of the manuscript.

This chapter was published as a paper in the Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN), 2019 [5]. The paper received the *best paper award* of the conference.

1.5 Conclusions and Outlook

1.5.1 Summary and Discussions of Contributions

In this section, we summarize the contributions of the different building blocks outlined in the thesis. We compare to related work and discuss the limitations of our work and possible future extensions.

A. Orchestra

We address the challenge of bringing TSCH (Time Slotted Channel Hopping MAC) to dynamic networks. We focus on low-power IPv6 and RPL networks and introduce Orchestra. Orchestra integrates TSCH in the IPv6 stack, extracts topology information from RPL and uses a rule-based approach to create communication schedules that fit the application traffic pattern while saving power by turning the radio off when no communication is taking place.

TSCH provides a reliable communication substrate and RPL provides the autonomous operation with self-forming and self-fixing abilities. The key idea is to provision a set of slots for different traffic planes and to define the slots in such a way that they can be automatically installed/removed as the routing topology evolves. This scheme allows Orchestra to build non-planned networks that support random traffic patterns while exploiting the robustness of TSCH.

Discussion Orchestra is significantly more reliable – by two orders of magnitude in our experiments – than state of the art asynchronous low-power routing while achieving a similar latency-energy balance. Compared to concurrent transmissions, Orchestra has the advantage to support random-access traffic, which makes it suitable for non-deterministic low-power IPv6 applications. Compared to centralized and decentralized scheduling solutions, Orchestra differs in that nodes compute their own schedule locally and autonomously, based on routing-layer information. Therefore, Orchestra could be considered a third

[3] `https://github.com/iot-chalmers/BlueFlood`

scheduling option for 6TiSCH networks, or can potentially be used jointly with other schedulers for facilitating the communication and maintenance of optimized schedules.

Limitations and Possible Extensions Orchestra builds its schedules based on runtime topology information from the routing protocol. This means that the quality of its schedules depends on the robustness of the routing protocol. Moreover, the assignment of timeslots to nodes in these schedules depends on the node IDs (a hash of the node ID); thus, it requires either a pre-deployment configuration, as we do in the chapter, or the realization of a join service. Finally, the schedules are over-provisioned; thus, non-optimal. Nevertheless, Orchestra is still a viable compromise, and applications with higher requirements may use it as a bootstrap method for establishing optimal schedules or flow reservations.

Summary To the best of our knowledge, Orchestra is the first distributed solution which does not require negotiation between nodes. We implement Orchestra in Contiki and demonstrate the practicality of Orchestra and quantify its benefits through extensive evaluation in simulation and two testbeds utilizing two hardware platforms. Orchestra reduces or even eliminates network contention. In long-running experiments of up to 72 hours, we show that Orchestra achieves end-to-end delivery ratios of over 99.99%. Compared to RPL in asynchronous low-power listening networks, Orchestra improves reliability by two orders of magnitude with a loss rate of 10^{-4} vs. 10^{-2}, while keeping a good latency-energy balance with twice the energy budget: 1.4% duty cycle for 0.5-second latency vs. 0.8% duty cycle for ContikiMAC.

To conclude, our first approach, Orchestra, builds on top of the TSCH MAC layer and utilizes RPL to build and maintain the mesh network. Orchestra's integration with the IP stack fits the general-purpose applications and supports IoT connectivity out-of-the-box at the cost of code complexity and RPL signaling overhead for maintaining the routing topology. Next, we discuss how concurrent transmissions support low-latency and maintenance-free network flooding.

B. A^2– Agreement in the Air and Synchrotron

We address low-latency and reliable consensus in low-power wireless networks. We argue that new approaches to concurrent transmissions, such as Glossy and Chaos, combined with a slotted architecture and frequency diversity, as done in TSCH for example, are key enablers for such protocols. We present A^2: Agreement in the Air, a system that brings distributed consensus to low-power multihop networks. A^2 introduces Synchrotron, a synchronous transmissions

kernel that builds a robust mesh by exploiting the capture effect, frequency hopping with parallel channels, and link-layer security. A^2 builds on top of this reliable base layer and enables the two- and three-phase commit protocols, as well as network services such as group membership, hopping sequence distribution, and re-keying.

Discussion Overall, synchronous transmissions enable low-latency network-wide communication. In contrast to A^2 and Chaos, nodes in Glossy always need to transmit the same data packet to enable non-destructive interference. As a result, they cannot exploit spatial diversity as Chaos and A^2 do. Moreover, their tight timing requirements of non-destructive interference limit the options of in-network processing of data. A^2 builds on these results and provides advanced communication primitives. While consensus has been studied in opportunistic and ad-hoc networks [11, 24], practical approaches are limited to single-hop networking [13, 27, 115] or have a latency of several seconds [41]. In contrast, A^2 builds on 2PC and 3PC to provide network-wide transactions in WSN, over hundreds of nodes, at low-latency and high reliability.

Limitations and Possible Extensions A^2 inherits Chaos packet size and bit-flag limitations as it requires at least one bit per node. A workaround could be possible by investigating data compression mechanisms, which in turn add to the computational overhead of A^2. Moreover, it inherits the limitations of the capture effect; namely, scalability and technology dependence: (i) The link quality degrades with the increase of the number of concurrent transmissions, and (ii) the utility of the capture effect varies with physical-layer design, *e.g.*, modulation, timing, and error-correction mechanisms. For example, it was shown that the capture-effect performance is questionable over WiFi [55] and Bluetooth [122]. Nevertheless, A^2 presents a viable design for 802.15.4 networks with hundreds of nodes. Besides, point-to-point communications are costly as A^2 builds on all-to-all primitives. In other words, to perform a point-to-point message delivery in A^2, we would need network-wide dissemination. Moreover, supporting random traffic patterns is costly since A^2 builds on periodic operation. For example, A^2 would need to either waste energy waking up often to check if there are events to serve or would adopt a low wakeup frequency to save energy at the cost of the high latency. Therefore, new mechanisms are needed to support an efficient event-triggered operation, as done in Crystal [65, 66], for example. Finally, A^2 is not standard-compliant. However, we conjecture that it is feasible to embed A^2 mechanisms over TSCH as concurrent transmission links are simply shared transmission slots in a TSCH schedule.

Summary We evaluate A^2 on four public testbeds with different deployment densities and sizes. Our extensive experimental evaluation shows that A^2 (i) is highly reliable, achieving zero losses over millions of data-points; (ii) achieves

low power and low latency, *e.g.*, A^2 requires only 475 ms to complete a two-phase commit over 180 nodes with a duty cycle of 0.5% for 1-minute intervals; and (iii) enables network-wide agreement, with different consistency/liveness tradeoffs: For example, the two-phase commit ensures transaction consistency in A^2 while the three-phase commit provides liveness at the expense of inconsistency under specific failure scenarios.

To conclude, A^2 builds on synchronous flooding and in-network processing methods, and extends them with time slotting and channel hopping. Synchronous flooding gives a reliable and autonomous communication substrate since flooding is a greedy strategy, and does not need to maintain the network topology. In-network processing enhances the low latency of the communication, as it aggregates data and executes the protocol logic while communicating. To this end, A^2 enables the realization of reliable consensus services. However, it does not address fault-tolerance which is paramount in practical solutions. Next, we discuss how to support fault-tolerant consensus over low-power wireless networks.

C. Wireless Paxos

We present Wireless Paxos, a fault-tolerant, network-wide consensus primitive that builds on top of concurrent transmissions to offer low-latency and reliable consensus in low-power wireless networks. While established designs of consensus protocols like Paxos are unfit for low-power wireless deployments due to their dependence on unicast communications, Wireless Paxos fills this gap by showing that Paxos can be expressed as a many-to-many communication scheme.

Discussion Most consensus solutions rely on routing or are limited to single-hop. In contrast, Wireless Paxos co-designs Paxos with the lower layers of the network stack to provide an efficient and low-latency consensus primitive for low-power wireless networks. We do not rely on any routing but utilize concurrent transmissions to communicate in multihop networks. While our A^2 provides an implementation of 2/3PC using concurrent transmissions [6], Wireless Paxos reuses the transmission kernel introduced by A^2, but the consensus primitives differ. A^2 handles failures by delaying the decision (consistency over availability), while Paxos handles failures through majorities.

Limitations and Possible Extensions We inherit the limitations of A^2, *e.g.*, the limited scalability and the need for one flag bit per node, and the limitations of Paxos, *e.g.*, requiring a majority. While we design Wireless Paxos to utilize the group communication and in-network processing features of A^2, we do not consider potential alternative designs that are better suited to the

partially synchronous nature of A^2. For example, with the support of reliable ordered multicasts over A^2, we can potentially design a simpler consensus protocol with a series of multicasts. An inspiring example protocol is NoPaxos [89] which is designed for data centers with programmable switches.

Summary By co-designing the Paxos consensus primitive along with the lower layers of the network stack and concurrent transmissions, we offer a highly reliable, fault-tolerant and low-latency consensus. We experimentally demonstrate that Wireless Paxos: (i) guarantees that at most one value can be agreed upon; (ii) provides consensus between 188 nodes in a testbed in 289 ms, and (iii) stays consistent under injected failures.

To conclude, Wireless Paxos and A^2 show how to build efficient and reliable network-wide consensus services. Both of them utilize concurrent transmission methods over IEEE 802.15.4 to achieve that. However, we miss a discussion of how concurrent transmissions (CT) apply to other low-power technologies. Next, we discuss supporting CT over Bluetooth.

D. BlueFlood

We address concurrent transmission over Bluetooth and study its feasibility in a controlled environment. Based on the feasibility study, we build BlueFlood; an efficient and reliable network flooding protocol inspired by Glossy.

Discussion Concurrent transmission protocols such as Glossy [42], LWB [40], Splash [28], Choco [136], Crystal [65], CXFS [18], Sparkle [156], Sleeping Beauty [126] and others [16, 68, 70, 127, 155] enable low-latency network-wide communication. While none of the related protocols support Bluetooth, the concepts are generally extendable to other technologies given that concurrent transmissions are supported. BlueFlood builds on these results to bring efficient network flooding to Bluetooth mesh networks.

Limitations and Possible Extensions The scope of the feasibility analysis of the concurrent transmissions in this chapter is limited to experimental evaluation. Further studies with physical layer modeling of concurrent transmissions are paramount. Moreover, the experimental evaluation of the BlueFlood protocol is limited to a single small testbed. Besides, the reliability of BlueFlood can be improved to achieve a loss rate below 10^{-5} as required for mission-critical applications. Finally, a study of the bit-error pattern in concurrent transmissions could lead to a viable forward error correction mechanism to improve reliability.

Summary Our experimental evaluation shows that: (i) although CT is more fragile over Bluetooth PHY than it is over 802.15.4, it is a viable communication strategy for network-wide dissemination; (ii) BlueFlood achieves data

dissemination with high reliability, low power, and low latency; (iii) the choice of the transmission mode (125 Kbps – 2 Mbps) provides a tradeoff between reliability, energy, and latency; and (iv) BlueFlood floods can be received on unmodified off-the-shelf Bluetooth-capable devices, *e.g.*, smartphones, and laptops.

1.5.2 Possible Future Directions

In this thesis, we investigate complementary approaches to build highly reliable, low power and low latency autonomous wireless networks. Our solutions run without any central scheduling entity or schedule negotiation and provide high-reliability communication. Moreover, we present protocols that address low-latency and reliable fault-tolerant consensus in low-power wireless networks as well. Finally, we show the feasibility of low-latency and reliable data dissemination over multihop Bluetooth mesh networks. Admittedly, while we investigate certain aspects of the aforementioned topics; we leave many open doors, and we realize other important aspects that we overlooked through our journey. In this section, we highlight some directions that are viable candidates for impactful future work.

A. Thorough Understanding of Concurrent Transmissions

In our work, we build protocols based on concurrent transmissions (CT), as CT enables efficient and reliable connectivity services. However, we only evaluate CT feasibility empirically over two physical layers; 802.15.4 and Bluetooth. Having a deeper understanding of CT would first strengthen our view of its strengths and limitations, Second, it would potentially uncover specialized techniques to enhance its resilience. For example, if we can uncover specific error patterns, we could design error correction techniques to specifically combat them. Moreover, since CT has a potential of improving protocol performance, we need to understand its applicability over other physical layers, *e.g.*, 5G-NBIoT and WiFi. While there have been studies on CT over WiFi [55], more can be done to investigate what features or techniques we can use to enhance CT quality over WiFi, for example.

B. Investigating Scheduling Mechanisms

While we have shown viable approaches for autonomously scheduled communication in Orchestra [34] and A^2 [6], both are suboptimal. To have nearly optimal results, communication scheduling shall be employed. The challenge

with scheduling is the high overhead for communicating and updating a schedule that adapts to the continuously changing wireless medium. However, with a careful balance of centrally devised, autonomous and peer-negotiated techniques, it is potentially possible to come up with a solution that outperforms both the slow classically scheduled and the suboptimal autonomously scheduled techniques.

C. Investigating Specialized Consensus Protocols

While we have demonstrated two alternative protocols for achieving strong consensus (with two-phase commit in A^2 [6]), and fault-tolerant eventual consistency (with Wireless Paxos over A^2 [112]), we still have space to potentially achieve higher performance by carefully designing a consensus protocol that fully utilizes the networking protocols' features. The aforementioned protocols make use of the group communication feature and in-network processing. While these consensus protocols, *i.e.*, 2PC and Paxos, are built for asynchronous or eventually synchronous systems, they are not optimal for partially synchronous systems that are synchronized and have access to a time reference. For example, with the support of reliable ordered multicasts over A^2, we can potentially design a simpler consensus protocol with a series of multicasts. Besides, we motivate our research with factory automation and UAV maneuver coordination, but due to the complexity of these systems and the lack of time, we do not attempt to engineer solutions that practically solve these problems. It would be interesting to investigate the application of distributed consensus protocols to solve these problems.

References

[1] ABI Research. *Installed Base of Bluetooth-enabled Devices Worldwide in 2012 and 2018*. Statista - The Statistics Portal. Aug. 2013. URL: `https://www.statista.com/statistics/283638/installed-base-forecast-bluetooth-enabled-devices-2012-2018/` (cit. on p. 38).

[2] J. Åkerberg, M. Gidlund, and M. Björkman. "Future research challenges in wireless sensor and actuator networks targeting industrial automation." In: *IEEE International Conference on Industrial Informatics*. July 2011 (cit. on pp. 3–5).

[3] Johan Åkerberg, Mikael Gidlund, Tomas Lennvall, Krister Landerns, and Mats Björkman. "Design Challenges and Objectives in Industrial Wireless Sensor Networks." In: *Industrial Wireless Sensor Networks: Applications, Protocols, and Standards*. 2013 (cit. on pp. 6, 32).

[4] Beshr Al Nahas, Simon Duquennoy, Venkatraman Iyer, and Thiemo Voigt. “Low-Power Listening Goes Multi-Channel.” In: *Proceedings of the Conference Distributed Computing in Sensor Systems (DCOSS)*. 2014 (cit. on pp. 23, 27).

[5] Beshr Al Nahas, Simon Duquennoy, and Olaf Landsiedel. “Concurrent Transmissions for Multi-Hop Bluetooth 5.” In: *Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN)*. 2019 (cit. on pp. 19, 38, 40).

[6] Beshr Al Nahas, Simon Duquennoy, and Olaf Landsiedel. “Network-wide Consensus Utilizing the Capture Effect in Low-power Wireless Networks.” In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2017 (cit. on pp. 32, 35, 37, 38, 43, 45, 46).

[7] Jude Allred, Ahmad Bilal Hasan, Saroch Panichsakul, William Pisano, Peter Gray, Jyh Huang, Richard Han, Dale Lawrence, and Kamran Mohseni. “SensorFlock: An Airborne Wireless Sensor Network of Micro-air Vehicles.” In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2007 (cit. on pp. 5, 32).

[8] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. “Basic Concepts and Taxonomy of Dependable and Secure Computing.” In: *IEEE Transactions on Dependable and Secure Computing* 1 (Jan. 2004) (cit. on p. 6).

[9] B. Kempke et al. “SurePoint: Exploiting Ultra Wideband Flooding and Diversity to Provide Robust, Scalable, High-Fidelity Indoor Localization.” In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2016 (cit. on pp. 24, 38).

[10] N. Baccour, A. Koubâa, C. Noda, H. Fotouhi, M. Alves, H. Youssef, M.A. Zúñiga, C.A. Boano, K. Römer, D. Puccinelli, et al. *Radio Link Quality Estimation in Low-Power Wireless Networks*. SpringerBriefs in Electrical and Computer Engineering. Springer International Publishing, 2013. ISBN: 9783319007748 (cit. on p. 8).

[11] Abdulkader Benchi and Pascale Launay. “Solving Consensus in Opportunistic Networks.” In: *ICDCN*. 2015. ISBN: 9781450329286. DOI: 10.1145/2684464.2684479 (cit. on pp. 25, 42).

[12] Bluetooth SIG. *Bluetooth 5 Core Specifications*. 2016. URL: https://www.bluetooth.com/specifications/bluetooth-core-specification (cit. on pp. 11, 23).

[13] C. A. Boano, M. A. Zuniga, K. Römer, and T. Voigt. “JAG: Reliable and Predictable Wireless Agreement under External Radio Inter-

ference." In: *Proceedings of the IEEE Real-Time Systems Symposium (IEEE RTSS)*. 2012 (cit. on pp. 25, 42).

[14] Martin Bor and Utz Roedig. "LoRa Transmission Parameter Selection." In: *Proceedings of the Conference Distributed Computing in Sensor Systems (DCOSS)*. 2017 (cit. on p. 9).

[15] F. Borran, R. Prakash, and A. Schiper. "Extending Paxos/LastVoting with an Adequate Communication Layer for Wireless Ad Hoc Networks." In: *IEEE SRDS*. 2008 (cit. on pp. 25, 36).

[16] M. Brachmann, O. Landsiedel, and S. Santini. "Concurrent Transmissions for Communication Protocols in the Internet of Things." In: *Proceedings of the Conference on Local Computer Networks (IEEE LCN)*. 2016 (cit. on pp. 24, 44).

[17] Nicolas Burri, Pascal Von Rickenbach, and Roger Wattenhofer. "Dozer: Ultra-Low Power Data Gathering in Sensor Networks." In: *Proceedings of the Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*. 2007 (cit. on pp. 22, 29).

[18] D. Carlson, M. Chang, A. Terzis, Y. Chen, and O. Gnawali. "Forwarder Selection in Multi-transmitter Networks." In: *Proceedings of the Conference Distributed Computing in Sensor Systems (DCOSS)*. 2013 (cit. on pp. 24, 38, 44).

[19] Miguel Castro and Barbara Liskov. "Practical Byzantine Fault Tolerance." In: *Proceedings of the Symposium on Operating Systems Design & Implementation (USENIX OSDI)*. 1999 (cit. on p. 25).

[20] Marco Cattani, Carlo Alberto Boano, and Kay Römer. "An Experimental Evaluation of the Reliability of LoRa Long-Range Low-Power Wireless Communication." In: *Journal of Sensor and Actuator Networks* 2 (June 2017) (cit. on p. 9).

[21] M. Ceriotti et al. "Is there light at the ends of the tunnel? Wireless sensor networks for adaptive lighting in road tunnels." In: *Proceedings of the Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*. 2011 (cit. on pp. 5, 32).

[22] *Ericsson Mobility Report - On the pulse of the Networked Society*. Tech. rep. Ericsson, Stockholm, Sweden, June 2019. URL: `https://www.ericsson.com/mobility-report` (cit. on p. 1).

[23] Tushar D. Chandra, Robert Griesemer, and Joshua Redstone. "Paxos Made Live: An Engineering Perspective." In: *ACM PODC*. 2007. DOI: `10.1145/1281100.1281103`. URL: `http://doi.acm.org/10.1145/1281100.1281103` (cit. on p. 36).

[24] Gregory Chockler, Murat Demirbas, Seth Gilbert, Calvin Newport, and Tina Nolte. “Consensus and Collision Detectors in Wireless Ad Hoc Networks.” In: *ACM PODC*. 2005 (cit. on pp. 25, 42).

[25] Pablo Corbalán and Gian Pietro Picco. “Concurrent Ranging in Ultra-wideband Radios: Experimental Evidence, Challenges, and Opportunities.” In: *Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN)*. 2018 (cit. on pp. 24, 38).

[26] George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. *Distributed Systems: Concepts and Design*. 5th. USA: Addison-Wesley Publishing Company, 2011. ISBN: 0132143011 (cit. on p. 12).

[27] M. Demirbas, O. Soysal, and M. Hussain. “TRANSACT: A Transactional Framework for Programming Wireless Sensor/Actor Networks.” In: *Proceedings of the Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*. 2008 (cit. on pp. 25, 42).

[28] Manjunath Doddavenkatappa, Mun Choon Chan, and Ben Leong. “Splash: Fast Data Dissemination with Constructive Interference in Wireless Sensor Networks.” In: *Proceedings of the Symposium on Networked Systems Design & Implementation (USENIX NSDI)*. 2013 (cit. on pp. 24, 44).

[29] Lance Doherty, William Lindsay, and Jonathan Simon. “Channel-Specific Wireless Sensor Network Path Data.” In: *Proceedings of the IEEE International Conference on Computer Communications and Networks (IEEE ICCCN)*. 2007 (cit. on pp. 21, 22, 29).

[30] Wan Du, Jansen Christian Liando, Huanle Zhang, and Mo Li. “When Pipelines Meet Fountain: Fast Data Dissemination in Wireless Sensor Networks.” In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2015 (cit. on p. 25).

[31] Wenliang Du, Jing Deng, Yunghsiang S. Han, Pramod K. Varshney, Jonathan Katz, and Aram Khalili. “A Pairwise Key Predistribution Scheme for Wireless Sensor Networks.” In: *ACM Transactions on Information and System Security (TISSEC)* (2005) (cit. on p. 32).

[32] Adam Dunkels. *The ContikiMAC Radio Duty Cycling Protocol*. Tech. rep. T2011:13. Swedish Institute of Computer Science, 2011 (cit. on p. 36).

[33] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. “Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors.” In: *Proceedings of the Conference on Local Computer Networks (IEEE LCN)*. 2004 (cit. on p. 31).

[34] Simon Duquennoy, Beshr Al Nahas, Olaf Landsiedel, and Thomas Watteyne. “Orchestra: Robust Mesh Networks Through Autonomously

Scheduled TSCH." In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2015 (cit. on pp. 27, 29, 32, 45).

[35] Simon Duquennoy, Atis Elsts, Beshr Al Nahas, and George Oikonomou. "TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation." In: *Proceedings of the Conference Distributed Computing in Sensor Systems (DCOSS)*. 2017 (cit. on pp. 25, 33).

[36] Simon Duquennoy, Olaf Landsiedel, and Thiemo Voigt. "Let the Tree Bloom: Scalable Opportunistic Routing with ORPL." In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2013 (cit. on pp. 22, 29).

[37] Simon Duquennoy, Fredrik Österlind, and Adam Dunkels. "Lossy Links, Low Power, High Throughput." In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2011 (cit. on pp. 25, 27).

[38] Prabal Dutta, Stephen Dawson-Haggerty, Yin Chen, Chieh-Jan Mike Liang, and Andreas Terzis. "Design and Evaluation of a Versatile and Efficient Receiver-Initiated Link Layer for Low-Power Wireless." In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2010 (cit. on p. 24).

[39] Atis Elsts et al. "Adaptive Channel Selection in IEEE 802.15.4 TSCH Networks." In: *Global Internet of Things Summit*. 2017 (cit. on p. 23).

[40] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. "Low-Power Wireless Bus." In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2012 (cit. on pp. 22, 24, 26, 27, 29, 33, 44).

[41] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. "Virtual Synchrony Guarantees for Cyber-physical Systems." In: *Proceedings of the IEEE International Symposium on Reliable Distributed Systems (IEEE SRDS)*. 2013 (cit. on pp. 25, 42).

[42] Federico Ferrari et al. "Efficient Network Flooding and Time Synchronization with Glossy." In: *Proceedings of the Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*. 2011 (cit. on pp. 11, 19, 20, 24, 28, 33, 38, 44).

[43] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. "Impossibility of Distributed Consensus with One Faulty Process." In: *J. ACM* (1985). ISSN: 0004-5411. DOI: 10.1145/3149.214121. URL: http://doi.acm.org/10.1145/3149.214121 (cit. on pp. 10, 35).

[44] HART Communication Foundation. *WirelessHART Specification 75: TDMA Data-Link Layer*. HCF_SPEC-75. HART Communication Foundation, 2008 (cit. on pp. 21, 29).

[45] Guillermo Gastón Lorente, Bart Lemmens, Matthias Carlier, An Braeken, and Kris Steenhaut. "BMRF: Bidirectional Multicast RPL Forwarding." In: *Ad Hoc Netw.* 54 (Jan. 2017) (cit. on p. 26).

[46] M. Gidlund, S. Han, et al. "Guest Editorial From Industrial Wireless Sensor Networks to Industrial Internet of Things." In: *IEEE Transactions on Industrial Informatics* 5 (May 2018) (cit. on p. 5).

[47] M. Gidlund, T. Lennvall, and J. Åkerberg. "Will 5G become yet another wireless technology for industrial automation?" In: *IEEE International Conference on Industrial Technology (ICIT)*. Mar. 2017 (cit. on p. 9).

[48] Seth Gilbert and Nancy Lynch. "Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-tolerant Web Services." In: *ACM SIGACT News* 2 (June 2002) (cit. on p. 10).

[49] Alasdair Gilchrist. *Industry 4.0: The Industrial Internet of Things*. 1st. Berkely, CA, USA: Apress, 2016. Chap. 13. ISBN: 9781484220467 (cit. on p. 2).

[50] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. "Collection Tree Protocol." In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2009 (cit. on pp. 16, 25).

[51] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. "Collection Tree Protocol." In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2009 (cit. on pp. 22, 29).

[52] António Gonga, Olaf Landsiedel, Pablo Soldati, and Mikael Johansson. "Revisiting Multi-channel Communication to Mitigate Interference and Link Dynamics in Wireless Sensor Networks." In: *Proceedings of the Conference Distributed Computing in Sensor Systems (DCOSS)*. 2012 (cit. on p. 8).

[53] Jim Gray. "Notes on Data Base Operating Systems." In: *Operating Systems, An Advanced Course*. 1978 (cit. on pp. 11, 12, 25, 32, 33).

[54] Jim Gray and Leslie Lamport. "Consensus on Transaction Commit." In: MSR-TR-2003-96 (Jan. 2004). ACM Transactions on Database Systems 31, 1 (2006). URL: `https://www.microsoft.com/en-us/research/publication/consensus-on-transaction-commit/` (cit. on pp. 13, 14).

[55] F. Gringoli, R. Klose, M. Hollick, and N. Ali. "Making Wi-Fi Fit for the Tactile Internet: Low-Latency Wi-Fi Flooding Using Concurrent

Transmissions." In: *IEEE International Conference on Communications Workshops (ICC Workshops)*. May 2018 (cit. on pp. 42, 45).

[56] Bernhard Großwindhager, Michael Stocker, Michael Rath, Carlo Alberto Boano, and Kay Römer. "SnapLoc: An Ultra-fast UWB-based Indoor Localization System for an Unlimited Number of Tags." In: *Proceedings of the Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*. 2019 (cit. on p. 24).

[57] Mesh Working Group. *Bluetooth Specification: Mesh Profile*. Bluetooth SIG. 2019. URL: `https://www.bluetooth.com/specifications/mesh-specifications/` (cit. on p. 16).

[58] Domenico De Guglielmo, Beshr Al Nahas, Simon Deuqennoy, Thiemo Voigt, and Giuseppe Anastasi. "Analysis and experimental evaluation of IEEE 802.15.4e TSCH CSMA-CA Algorithm." In: PP (99 2016) (cit. on p. 22).

[59] G. Gupta and M. Younis. "Fault-tolerant clustering of wireless sensor networks." In: *Proceedings of the IEEE Conference on Wireless Communications and Networking (WCNC)*. 2003 (cit. on p. 25).

[60] Carsten Herrmann, Fabian Mager, and Marco Zimmerling. "Mixer: Efficient Many-to-All Broadcast in Dynamic Wireless Mesh Networks." In: *ACM SenSys*. 2018 (cit. on pp. 24, 26).

[61] IEEE. *802.15.4-2015: IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) 1: MAC sublayer*. Apr. 2016 (cit. on pp. 11, 16, 21, 23, 25, 29, 32, 33).

[62] IEEE. *802.15.4-2015: IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) 1: MAC sublayer*. 2016 (cit. on p. 28).

[63] ISA. *ISA-100.11a-2011 – Wireless Systems for Industrial Automation: Process Control and Related Applications*. ISA, 2011 (cit. on pp. 21, 29).

[64] Michael Isard. *Autopilot: Automatic Data Center Management*. Tech. rep. Microsoft, 2007 (cit. on p. 36).

[65] Timofei Istomin, Amy L. Murphy, Gian Pietro Picco, and Usman Raza. "Data Prediction + Synchronous Transmissions = Ultra-low Power Wireless Sensor Networks." In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2016 (cit. on pp. 24, 42, 44).

[66] Timofei Istomin, Matteo Trobinger, Amy L. Murphy, and Gian Pietro Picco. "Interference-Resilient Ultra-Low Power Aperiodic Data Collec-

tion." In: *Proceedings of the Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*. 2018 (cit. on pp. 24, 42).

[67] Yogesh G. Iyer, Shashidhar Gandham, and S. Venkatesan. "STCP: A Generic Transport Layer Protocol for Wireless Sensor Networks." In: *Proceedings of the IEEE International Conference on Computer Communications and Networks (IEEE ICCCN)*. 2005 (cit. on p. 25).

[68] J. Jeong et al. "Low-power and topology-free data transfer protocol with synchronous packet transmissions." In: *Proceedings of the Conference on Sensor, Mesh and Ad Hoc Communications and Networks (IEEE SECON)*. 2014 (cit. on pp. 24, 44).

[69] J. Nieminen et al. *IPv6 over Bluetooth Low Energy*. RFC 7668. 2015 (cit. on p. 23).

[70] J. Zhang et al. "RFT: Identifying Suitable Neighbors for Concurrent Transmissions in Point-to-Point Communications." In: *Proceedings of the Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (ACM MSWiM)*. 2015 (cit. on pp. 24, 44).

[71] Romain Jacob, Jonas Baechli, Reto Da Forno, and Lothar Thiele. "Synchronous Transmissions Made Easy: Design Your Network Stack with Baloo." In: *Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN)*. 2019 (cit. on p. 24).

[72] Shouling Ji, Jing (Selena) He, and Zhipeng Cai. "Data Gathering in Wireless Sensor Networks." In: *The Art of Wireless Sensor Networks*. Ed. by Habib M. Ammari. Signals and Communication Technology. Springer, 2014 (cit. on p. 25).

[73] Sokratis Kartakis, Babu D. Choudhary, Alexander D. Gluhak, Lambros Lambrinos, and Julie A. McCann. "Demystifying Low-power Wide-area Communications for City IoT Applications." In: *Proceedings of the ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization (ACM WiNTECH)*. 2016 (cit. on p. 9).

[74] Anna N. Kim, Fredrik Hekland, Stig Petersen, and Paula Doyle. "When HART goes wireless: Understanding and implementing the WirelessHART standard." In: *IEEE Conference on Emerging Technologies Factory Automation (ETFA)*. 2008 (cit. on p. 21).

[75] Sukun Kim, Rodrigo Fonseca, Prabal Dutta, Arsalan Tavakoli, David Culler, Philip Levis, Scott Shenker, and Ion Stoica. "Flush: A Reliable Bulk Transport Protocol for Multihop Wireless Networks." In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2007 (cit. on p. 25).

[76] JeongGil Ko, Joakim Eriksson, Nicolas Tsiftes, Stephen Dawson-Haggerty, Jean-Philippe Vasseur, Mathilde Durvy, Andreas Terzis, Adam Dunkels, and David Culler. "Industry: Beyond Interoperability: Pushing the Performance of Sensor Network IP Stacks." In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2011 (cit. on p. 29).

[77] Andreas Köpke. "Engineering a communication protocol stack to support consensus in sensor networks." PhD thesis. TU Berlin, 2012 (cit. on p. 25).

[78] B. Krishnamachari and S. Iyengar. "Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks." In: *IEEE Transactions on Computers* 53.3 (2004) (cit. on p. 25).

[79] Ajay D. Kshemkalyani and Mukesh Singhal. *Distributed Computing: Principles, Algorithms, and Systems*. New York, NY, USA: Cambridge University Press, 2011. ISBN: 0521189845 (cit. on p. 12).

[80] Leslie Lamport. "Fast Paxos." In: *Distributed Computing* 19.2 (2006) (cit. on p. 25).

[81] Leslie Lamport. "Paxos Made Simple." In: *SIGACT* 32 (2001). URL: `https://www.microsoft.com/en-us/research/publication/paxos-made-simple/` (cit. on pp. 11, 25, 35).

[82] Leslie Lamport. "The part-time parliament." In: *ACM TOCS* 16.2 (1998). DOI: `10.1145/279227.279229`. URL: `https://doi.org/10.1145/279227.279229` (cit. on pp. 11–13, 25, 35, 36).

[83] Leslie Lamport and Mike Massa. "Cheap Paxos." In: *IEEE/IFIP DSN*. 2004. ISBN: 0-7695-2052-9. URL: `http://dl.acm.org/citation.cfm?id=1009382.1009745` (cit. on p. 25).

[84] Olaf Landsiedel, Federico Ferrari, and Marco Zimmerling. "Chaos: Versatile and Efficient All-to-All Data Sharing and In-Network Processing at Scale." In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2013 (cit. on pp. 19–21, 24, 26–28, 33).

[85] Olaf Landsiedel, Euhanna Ghadimi, Simon Duquennoy, and Mikael Johansson. "Low Power, Low Delay: Opportunistic Routing meets Duty Cycling." In: *Proceedings of the Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*. 2012 (cit. on pp. 22, 29).

[86] Jean-Claude Laprie. "Dependable Computing: Concepts, Limits, Challenges." In: *Proceedings of the International Conference on Fault-tolerant Computing (FTCS)*. IEEE Computer Society, 1995 (cit. on p. 6).

[87] K. Leentvaar and J. Flint. “The Capture Effect in FM Receivers.” In: *IEEE Transactions on Communications* 24.5 (1976) (cit. on pp. 18, 19, 21, 23).

[88] T. Lennvall, M. Gidlund, and J. Åkerberg. “Challenges when bringing IoT into industrial automation.” In: *IEEE AFRICON*. Sept. 2017 (cit. on pp. 3–5).

[89] Jialin Li, Ellis Michael, Naveen Kr. Sharma, Adriana Szekeres, and Dan R. K. Ports. “Just Say NO to Paxos Overhead: Replacing Consensus with Network Ordering.” In: *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. USENIX Association, 2016. URL: `https://www.usenix.org/conference/osdi16/technical-sessions/presentation/li` (cit. on p. 44).

[90] W. Liang, M. Zheng, J. Zhang, H. Shi, H. Yu, Y. Yang, S. Liu, W. Yang, and X. Zhao. “WIA-FA and Its Applications to Digital Factory: A Wireless Network Solution for Factory Automation.” In: *Proceedings of the IEEE* 6 (June 2019) (cit. on pp. 3, 4).

[91] Chun-Hao Liao, Yuki Katsumata, Makoto Suzuki, and Hiroyuki Morikawa. “Revisiting the So-Called Constructive Interference in Concurrent Transmission.” In: *Proceedings of the Conference on Local Computer Networks (IEEE LCN)*. 2016 (cit. on pp. 19, 20, 24).

[92] Chun-Hao Liao, Makoto Suzuki, and Hiroyuki Morikawa. “Poster Abstract: Toward Robust Concurrent Transmission for sub-GHz Non-DSSS Communication.” In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2016 (cit. on p. 24).

[93] Hai Liu, Amiya Nayak, and Ivan Stojmenović. “Fault-Tolerant Algorithms/Protocols in Wireless Sensor Networks.” In: *Guide to Wireless Sensor Networks*. Ed. by Subhas Chandra Misra, Isaac Woungang, and Sudip Misra. 2009 (cit. on p. 25).

[94] X. Luo, M. Dong, and Y. Huang. “On distributed fault-tolerant detection in wireless sensor networks.” In: *IEEE Transactions on Computers* 55.1 (2006) (cit. on p. 25).

[95] Parisa Jalili Marandi, M. Primi, N. Schiper, and F. Pedone. “Ring Paxos: A high-throughput atomic broadcast protocol.” In: *IEEE/IFIP DSN*. 2010 (cit. on pp. 25, 36).

[96] Piergiuseppe Di Marco, Per Skillermark, Anna Larmo, and Pontus Arvidson. *Bluetooth Mesh Networking*. Ericsson AB. 2017. URL: `https://www.ericsson.com/assets/local/publications/white-papers/wp-bluetooth-mesh_ver2_171115-c2.pdf` (cit. on p. 16).

[97] J. -P. Martin and L. Alvisi. “Fast Byzantine Consensus.” In: *IEEE Trans. on Dependable and Secure Computing* 3.3 (2006) (cit. on p. 35).

[98] J. Martocci, P. De Mil, N. Riou, and W. Vermeylen. *Building Automation Routing Requirements in Low-Power and Lossy Networks*. RFC 5867 (Proposed Standard). June 2010 (cit. on p. 4).

[99] Mobashir Mohammad and Mun Choon Chan. "Codecast: Supporting Data Driven In-network Processing for Low-power Wireless Sensor Networks." In: *Proceedings of the Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*. 2018 (cit. on p. 24).

[100] H. Moniz, N. F. Neves, and M. Correia. "Byzantine Fault-Tolerant Consensus in Wireless Ad Hoc Networks." In: *IEEE Trans. on Mobile Computing* 12.12 (2013) (cit. on p. 25).

[101] A. Morell, X. Vilajosana, J. L. Vicario, and T. Watteyne. "Label switching over IEEE802.15.4e networks." In: *Transactions on Emerging Telecommunications Technologies* 24.5 (Aug. 2013) (cit. on p. 22).

[102] David Moss and Philip Levis. *BoX-MACs: Exploiting Physical and Link Layer Boundaries in Low-Power Networking*. Tech. rep. SING-08-00. Stanford, 2008 (cit. on pp. 27, 36).

[103] V. Navda, A. Bohra, S. Ganguly, and D. Rubenstein. "Using Channel Hopping to Increase 802.11 Resilience to Jamming Attacks." In: *Proceedings of the Conference on Computer Communications (IEEE INFOCOM)*. 2007 (cit. on pp. 5, 32).

[104] N. Nowdehi, A. Lautenbach, and T. Olovsson. "In-Vehicle CAN Message Authentication: An Evaluation Based on Industrial Criteria." In: *IEEE Vehicular Technology Conference (VTC Fall)*. Sept. 2017 (cit. on p. 1).

[105] Tony O'donovan et al. "The GINSENG System for Wireless Monitoring and Control: Design and Deployment Experiences." In: *ACM Transactions on Senensor Networks (ACM TOSN)* 10.1 (2013) (cit. on pp. 5, 32).

[106] G. Oikonomou and I. Phillips. "Stateless multicast forwarding with RPL in 6LowPAN sensor networks." In: *IEEE International Conference on Pervasive Computing and Communications, Workshops*. 2012 (cit. on p. 26).

[107] Diego Ongaro and John Ousterhout. "In Search of an Understandable Consensus Algorithm." In: *USENIX ATC*. 2014. ISBN: 978-1-931971-10-2 (cit. on pp. 11, 25, 35).

[108] Jeongyeup Paek and Ramesh Govindan. "RCRT: Rate-controlled Reliable Transport Protocol for Wireless Sensor Networks." In: *ACM Transactions on Senensor Networks (ACM TOSN)* 7.3 (2010) (cit. on p. 25).

[109] Maria Rita Palattella, Nicola Accettura, Mischa Dohler, Luigi Alfredo Grieco, and Gennaro Boggia. "Traffic Aware Scheduling Algorithm for

reliable low-power multi-hop IEEE 802.15.4e networks." In: *Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. 2012 (cit. on p. 22).

[110] Maria Rita Palattella, Nicola Accettura, Luigi Alfredo Grieco, Gennaro Boggia, Mischa Dohler, and Thomas Engel. "On Optimal Scheduling in Duty-Cycled Industrial IoT Applications using IEEE802.15.4e TSCH." In: *IEEE Sensors Journal* (2013) (cit. on p. 22).

[111] K. Pister, P. Thubert, S. Dwars, and T. Phinney. *Industrial Routing Requirements in Low-Power and Lossy Networks*. RFC 5673 (Proposed Standard). Oct. 2009 (cit. on p. 3).

[112] Valentin Poirot, Beshr Al Nahas, and Olaf Landsiedel. "Paxos Made Wireless: Consensus in the Air." In: *Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN)*. 2019 (cit. on pp. 35, 37, 38, 46).

[113] Wolf-Bastian Pöttner, Hans Seidel, James Brown, Utz Roedig, and Lars Wolf. "Constructing Schedules for Time-Critical Data Delivery in Wireless Sensor Networks." In: *ACM Transactions on Senensor Networks (ACM TOSN)* 10.3 (2014) (cit. on pp. 22, 29).

[114] Daniele Puccinelli, Silvia Giordano, Marco Zuniga, and Pedro José Marrón. "Broadcast-free Collection Protocol." In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2012 (cit. on p. 22).

[115] Q. Wang and others. *6top Protocol (6P)*. IETF draft-ietf-6tisch-6top-protocol-04, WiP. 2017 (cit. on pp. 25, 42).

[116] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic. "Cyber-physical systems: The next computing revolution." In: *Design Automation Conference*. June 2010 (cit. on p. 1).

[117] Sumit Rangwala, Ramakrishna Gummadi, Ramesh Govindan, and Konstantinos Psounis. "Interference-aware Fair Rate Control in Wireless Sensor Networks." In: *SIGCOMM Computer Communication Review* 36.4 (2006) (cit. on p. 25).

[118] Jun Rao, Eugene J. Shekita, and Sandeep Tata. "Using Paxos to build a scalable, consistent, and highly available datastore." In: *VLDB Endowment* 4.4 (2011). URL: `http://dl.acm.org/citation.cfm?doid=1938545.1938549` (cit. on p. 36).

[119] V. S. Rao, M. Koppal, R. V. Prasad, T. V. Prabhakar, C. Sarkar, and I. Niemegeers. "Murphy loves CI: Unfolding and improving constructive interference in WSNs." In: *Proceedings of the Conference on Computer Communications (IEEE INFOCOM)*. 2016 (cit. on p. 23).

[120] W. Ren, R. W. Beard, and E. M. Atkins. “Information consensus in multivehicle cooperative control.” In: *IEEE Control Systems Magazine* 2 (Apr. 2007) (cit. on p. 33).

[121] Matthias Ringwald and Kay Römer. “BitMAC: A deterministic, collision-free, and robust MAC protocol for sensor networks.” In: *Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN)*. 2005 (cit. on pp. 23, 24).

[122] Coen Roest. “Enabling the Chaos Networking Primitive on Bluetooth LE.” MA thesis. Chalmers Univ. of Tech. and TU Delft, 2015 (cit. on pp. 24, 42).

[123] Raúl Rondón, Mikael Gidlund, and Krister Landernäs. “Evaluating Bluetooth Low Energy Suitability for Time-Critical Industrial IoT Applications.” In: *International Journal of Wireless Information Networks* 3 (Sept. 2017) (cit. on pp. 3, 4).

[124] A. Saifullah, You Xu, Chenyang Lu, and Yixin Chen. “Real-Time Scheduling for WirelessHART Networks.” In: *Proceedings of the IEEE Real-Time Systems Symposium (IEEE RTSS)*. 2010 (cit. on p. 22).

[125] Abusayeed Saifullah, Paras Babu Tiwari, Bo Li, Chenyang Lu, and Yixin Chen. *Accounting for Failures in Delay Analysis for WirelessHART Networks*. 2012 (cit. on p. 22).

[126] C. Sarkar, R. V. Prasad, R. T. Rajan, and K. Langendoen. “Sleeping Beauty: Efficient Communication for Node Scheduling.” In: *Proceedings of the IEEE International Conference on Mobile Ad-Hoc and Smart Systems (IEEE MASS)*. 2016 (cit. on pp. 24, 44).

[127] Chayan Sarkar. *LWB and FS-LWB implementation for Sky nodes using Contiki*. arXiv preprint. 2016. URL: `https://arxiv.org/pdf/1607.06622.pdf` (cit. on pp. 24, 44).

[128] Thomas Schmid, Prabal Dutta, and Mani B. Srivastava. “High-resolution, Low-power Time Synchronization an Oxymoron No More.” In: *Proceedings of the Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*. 2010 (cit. on p. 33).

[129] Markus Schuß, Carlo Alberto Boano, Manuel Weber, and Kay Uwe Römer. “A Competition to Push the Dependability of Low-Power Wireless Protocols to the Edge.” In: *Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN)*. 2017 (cit. on p. 23).

[130] *Short Range Devices (SRD) operating in the frequency range 25 MHz to 1 000 MHz*. ETSI EN 300-220. 2017 (cit. on p. 9).

[131] Silicon Labs. *Bluetooth Mesh Network Performance*. `https://www.silabs.com/documents/login/application-notes/an1137-bluetooth-mesh-network-performance.pdf`. 2018 (cit. on p. 16).

[132] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund. "Industrial Internet of Things: Challenges, Opportunities, and Directions." In: *IEEE Transactions on Industrial Informatics* 11 (Nov. 2018) (cit. on pp. 5–7).

[133] D. Skeen and M. Stonebraker. "A Formal Model of Crash Recovery in a Distributed System." In: *IEEE Transactions on Software Engineering* SE-9.3 (1983) (cit. on pp. 11, 12, 25, 33).

[134] Dongjin Son, Bhaskar Krishnamachari, and John Heidemann. "Experimental Study of Concurrent Transmission in Wireless Sensor Networks." In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2006 (cit. on p. 23).

[135] Michael Spörk, Carlo Alberto Boano, Marco Zimmerling, and Kay Römer. "BLEach: Exploiting the Full Potential of IPv6 over BLE in Constrained Embedded IoT Devices." In: *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 2017 (cit. on p. 23).

[136] M. Suzuki, Y. Yamashita, and H. Morikawa. "Low-Power, End-to-End Reliable Collection Using Glossy for Wireless Sensor Networks." In: *IEEE Vehicular Technology Conference (VTC Spring)*. 2013 (cit. on pp. 24, 44).

[137] T. Winter et al. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*. RFC 6550 (Proposed Standard). 2012 (cit. on pp. 16–18, 25, 32).

[138] Lei Tang, Yanjun Sun, Omer Gurewitz, and David B. Johnson. "EM-MAC: A Dynamic Multichannel Energy-efficient MAC Protocol for Wireless Sensor Networks." In: *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (ACM MobiHoc)*. 2011 (cit. on p. 23).

[139] X. Thubert (Ed.), T. Watteyne, R. Struik, and M. Richardson. *An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4e - draft-ietf-6tisch-architecture-06*. IETF Draft. Mar. 2015 (cit. on pp. 22, 30).

[140] Andrew Tinka, Thomas Watteyne, Kristofer S. J. Pister, and Alexandre M. Bayen. "A Decentralized Scheduling Algorithm for Time Synchronized Channel Hopping." In: *EAI Endorsed Transactions on Mobile Communications and Applications* 11.1 (2011) (cit. on p. 22).

[141] *Tmote sky: Ultra low power IEEE 802.15.4 compliant wireless sensor module*. Data sheet. San Francisco, CA 94105, 2006 (cit. on p. 7).

[142] Piercarlo Valdesolo. “Scientists Study Nomophobia — Fear of Being without a Mobile Phone.” In: *Scientific American*. Oct. 2015 (cit. on p. 1).

[143] M. C. Vuran and I. F. Akyildiz. “Error Control in Wireless Sensor Networks: A Cross Layer Analysis.” In: *IEEE/ACM Transactions on Networking* 4 (Aug. 2009) (cit. on p. 7).

[144] Brett Warneke, Matt Last, Brian Liebowitz, and Kristofer Pister. “Smart dust: communicating with a cubic-millimeter computer.” In: *Computer* (Feb. 2001) (cit. on p. 7).

[145] T. Watteyne, S. Lanzisera, A. Mehta, and K.S.J. Pister. “Mitigating Multipath Fading through Channel Hopping in Wireless Sensor Networks.” In: *Proceedings of the IEEE International Conference on Communications (IEEE ICC)*. 2010 (cit. on pp. 8, 17, 23, 29).

[146] Thomas Watteyne, Lance Doherty, Jonathan Simon, and Kris Pister. “Technical Overview of SmartMesh IP.” In: *Proceedings of the International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. 2013 (cit. on p. 22).

[147] Thomas Watteyne, Ankur Mehta, and Kris Pister. “Reliability Through Frequency Diversity: Why Channel Hopping Makes Sense.” In: *Proceedings of the ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN)*. 2009 (cit. on p. 8).

[148] Thomas Watteyne, Joy Weiss, Lance Doherty, and Jonathan Simon. “Industrial IEEE802.15.4e Networks: Performance and Trade-offs.” In: *Proceedings of the IEEE International Conference on Communications (IEEE ICC)*. 2015 (cit. on p. 22).

[149] M. Wilhelm, V. Lenders, and J. B. Schmitt. “On the Reception of Concurrent Transmissions in Wireless Sensor Networks.” In: *IEEE Transactions on Wireless Communications* (2014) (cit. on pp. 19, 24).

[150] “Wireless sensors replace cables and save tonnes of copper and plastic.” In: *Volvo Group News* (Nov. 18, 2016). URL: `https://www.volvogroup.com/en-en/news/2016/nov/wireless-sensors-replace-cables-and-save-tonnes-of-copper-and-plastic.html` (visited on 09/23/2019) (cit. on p. 1).

[151] Martin Woolley. *Bluetooth 5: Go Faster. Go Further*. Bluetooth SIG. 2018. URL: `https://www.bluetooth.com/bluetooth-technology/bluetooth5/bluetooth5-paper` (cit. on p. 15).

[152] Mao Ye, Chengfa Li, Guihai Chen, and J. Wu. “EECS: an energy efficient clustering scheme in wireless sensor networks.” In: *IEEE Inter-*

national Performance, Computing, and Communications Conference (PCCC). 2005 (cit. on p. 32).

[153] O. N. C. Yilmaz, Y. -. E. Wang, N. A. Johansson, N. Brahmi, S. A. Ashraf, and J. Sachs. "Analysis of ultra-reliable and low-latency 5G communication for a factory automation use case." In: *IEEE International Conference on Communication Workshop (ICCW)*. June 2015 (cit. on p. 5).

[154] Y. H. Yitbarek, K. Yu, J. Åkerberg, M. Gidlund, and M. Björkman. "Implementation and evaluation of error control schemes in Industrial Wireless Sensor Networks." In: *IEEE International Conference on Industrial Technology (ICIT)*. Feb. 2014 (cit. on p. 7).

[155] D. Yuan and M. Hollick. "Ripple: High-throughput, reliable and energy-efficient network flooding in wireless sensor networks." In: *Proceedings of the Symposium on a World of Wireless Mobile and Multimedia Networks (IEEE WoWMoM)*. 2015 (cit. on pp. 24, 44).

[156] Dingwen Yuan, Michael Riecker, and Matthias Hollick. "Making 'Glossy' Networks Sparkle: Exploiting Concurrent Transmissions for Energy Efficient, Reliable, Ultra-Low Latency Communication in Wireless Control Networks." In: *Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN)*. 2014 (cit. on pp. 24, 44).

[157] Pouria Zand, Arta Dilo, and Paul Havinga. "D-MSR: A Distributed Network Management Scheme for Real-Time Monitoring and Process Control Applications in Wireless Industrial Automation." In: *Sensors* 13.7 (2013) (cit. on p. 22).

[158] Haibo Zhang, Pablo Soldati, and Mikael Johansson. "Optimal Link Scheduling and Channel Assignment for Convergecast in Linear wirelessHART Networks." In: *Proceedings of the 7th International Conference on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*. 2009 (cit. on p. 22).

[159] Marco Zimmerling. "End-to-End Predictability and Efficiency in Low-Power Wireless Networks." Ph.D. Dissertation. ETH Zurich, July 2015 (cit. on p. 8).